INTERNATIONAL STANDARD

ISO/IEC 14496-3

Second edition
2001-12-15

AMENDMENT 2
2004-08-01

Information technology — Coding of audio-visual objects —

Part 3: **Audio**

AMENDMENT 2: Parametric coding for high-quality audio

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

AMENDEMENT 2: Codage paramétrique pour le codage audio de haute qualité

Reference number
ISO/IEC 14496-3:2001/Amd.2:2004(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 2 to ISO/IEC 14496-3:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Introduction

This document specifies the second Amendment to ISO/IEC 14496-3:2001. The document specifies the normative syntax of the 'Parametric Coding for High Quality Audio' tool SSC and the decoding process. An informative encoder description is given as well.

Information technology — Coding of audio-visual objects —

Part 3:
Audio

AMENDMENT 2: Parametric coding for high-quality audio

In ISO/IEC 14496-3:2001, Introduction, add:

MPEG-4 SSC, (SinuSoidal Coding) is a parametric coding tool that is capable of full bandwidth high quality audio coding. The coding tool dissects a monaural or stereo audio signal into a number of different objects that each can be parameterized efficiently and encoded at a low bit-rate. These objects are transients: representing dynamic changes in the temporal domain, sinusoids: representing deterministic components, and noise: representing components that do not have a clear temporal or spectral localisation. The fourth object, that is only relevant for stereo input signals, captures the stereo image. As the signal is represented in a parametric domain, independent, high quality pitch and tempo scaling are possible at low computational cost.

Amendment subpart 1

In Part 3: Audio, Subpart 1, in subclause 1.3 Terms and Definitions, add:

270. **SSC**: SinuSoidal Coding.

and increase the index-number of subsequent entries.

In Part 3: Audio, Subpart 1, in subclause 1.5.1.1 Audio object type definition, replace table 1.1 with the table below:

Table 1.1 – Audio object definition

Tools/ Modules Audio Object Type	gain control	block switching	window shapes - standard	window shapes – AAC LD	filterbank - standard	filterbank – SSR	TNS	LTP	intensity	coupling	MPEG-2 prediction	PNS	MS	SIAQ	FSS	upsampling filter tool	quantisation&coding - AAC	quantisation&coding - TwinVQ	quantisation&coding - BSAC	AAC ER Tools	ER payload syntax	EP Tool 1)	CELP	Silence Compression	HVXC	HVXC 4kbs VR	SA tools	SASBF	MIDI	HILN	TTSI	SBR	SSC	Remark	Object Type ID
Null																																		0
AAC main		X	X		X		X		X	X	X	X	X				X																2)	1
AAC LC		X	X		X		X		X	X		X	X				X																	2
AAC SSR	X	X	X			X	X		X	X		X	X				X																	3
AAC LTP		X	X		X		X	X	X	X		X	X				X																2)	4
SBR																																X			5
AAC Scalable		X	X		X		X	X	X			X	X	X	X	X	X																6)	6
TwinVQ		X	X		X		X	X					X					X																7
CELP																							X											8
HVXC																									X									9
(Reserved)																																		10
(Reserved)																																		11
TTSI																															X			12
Main synthetic																											X	X	X				3)	13
Wavetable synthesis																												X	X				4)	14
General MIDI																													X					15
Algorithmic Synthesis and Audio FX																											X							16
ER AAC LC		X	X		X		X		X			X	X				X			X	X	X												17
(Reserved)																																		18
ER AAC LTP		X	X		X		X	X	X			X	X				X			X	X	X											5)	19
ER AAC scalable		X	X		X		X		X			X	X	X	X	X	X			X	X	X											6)	20
ER TwinVQ		X	X		X		X						X					X			X	X												21
ER BSAC		X	X		X		X		X			X	X						X		X	X												22
ER AAC LD				X	X		X	X	X			X	X				X			X	X	X												23
ER CELP																					X	X	X	X										24
ER HVXC																					X	X			X	X								25
ER HILN																					X	X								X				26
ER Parametric																					X	X			X	X				X				27
SSC																																X		28
(Reserved)																																		29
(Reserved)																																		30
(Reserved)																																		31

In Part 3: Audio, Subpart 1, replace Table 1.2 (Audio Profiles definition) with the following table:

Table 1.2 – Audio Profiles definition

Audio Object Type	Main Audio Profile	Scalable Audio Profile	Speech Audio Profile	Synthetic Audio Profile	High Quality Audio Profile	Low Delay Audio Profile	Natural Audio Profile	Mobile Audio Internet-working Profile	AAC Profile	High Efficiency AAC Profile	Object Type ID
Null											0
AAC main	X						X				1
AAC LC	X	X			X		X		X	X	2
AAC SSR	X						X				3
AAC LTP	X	X			X		X				4
SBR										X	5
AAC Scalable	X	X			X		X				6
TwinVQ	X	X					X				7
CELP	X	X	X		X	X	X				8
HVXC	X	X	X			X	X				9
(reserved)											10
(reserved)											11
TTSI	X	X	X	X		X	X				12
Main synthetic	X			X							13
Wavetable synthesis											14
General MIDI											15
Algorithmic Synthesis and Audio FX											16
ER AAC LC					X		X	X			17
(reserved)											18
ER AAC LTP					X		X				19
ER AAC Scalable					X		X	X			20
ER TwinVQ							X	X			21
ER BSAC							X	X			22
ER AAC LD						X	X	X			23
ER CELP					X	X	X				24
ER HVXC						X	X				25
ER HILN							X				26
ER Parametric							X				27
SSC											28
(reserved)											29
(reserved)											30
(reserved)											31

In Part 3: Audio, Subpart 1, in subclause 1.6.2.1 AudioSpecificConfig, replace table 1.8 with the table below:

Table 1.8 – Syntax of AudioSpecificConfig()

Syntax	No. of bits	Mnemonic
AudioSpecificConfig ()		
{		
audioObjectType;	**5**	**uimsbf**
samplingFrequencyIndex;	**4**	**uimsbf**
if (samplingFrequencyIndex==0xf)		
samplingFrequency;	**24**	**uimsbf**
channelConfiguration;	**4**	**uimsbf**
sbrPresentFlag = -1;		
if (audioObjectType == 5) {		
extensionAudioObjectType = audioObjectType;		
sbrPresentFlag = 1;		
extensionSamplingFrequencyIndex;	**4**	**uimsbf**
if (extensionSamplingFrequencyIndex==0xf)		
extensionSamplingFrequency;	**24**	**uimsbf**
audioObjectType;	**5**	**uimsbf**
}		
else {		
extensionAudioObjectType = 0;		
}		
if (audioObjectType == 1 || audioObjectType == 2 ||		
audioObjectType == 3 || audioObjectType == 4 ||		
audioObjectType == 6 || audioObjectType == 7)		
GASpecificConfig();		
if (audioObjectType == 8)		
CelpSpecificConfig();		
if (audioObjectType == 9)		
HvxcSpecificConfig();		
if (audioObjectType == 12)		
TTSSpecificConfig();		
if (audioObjectType == 13 || audioObjectType == 14 ||		
audioObjectType == 15 || audioObjectType==16)		
StructuredAudioSpecificConfig();		
if (audioObjectType == 17 || audioObjectType == 19 ||		
audioObjectType == 20 || audioObjectType == 21 ||		
audioObjectType == 22 || audioObjectType == 23)		
GASpecificConfig();		
if (audioObjectType == 24)		
ErrorResilientCelpSpecificConfig();		
if (audioObjectType == 25)		
ErrorResilientHvxcSpecificConfig();		
if (audioObjectType == 26 || audioObjectType == 27)		
ParametricSpecificConfig();		
if (audioObjectType == 17 || audioObjectType == 19 ||		
audioObjectType == 20 || audioObjectType == 21 ||		
audioObjectType == 22 || audioObjectType == 23 ||		
audioObjectType == 24 || audioObjectType == 25 ||		
audioObjectType == 26 || audioObjectType == 27) {		
epConfig;	**2**	**uimsbf**
if (epConfig == 2 || epConfig == 3) {		
ErrorProtectionSpecificConfig();		
}		
if (epConfig == 3) {		

```
                directMapping;                                          1           uimsbf
                if ( ! directMapping ) {
                    /* tbd */
                }
            }
        }
        if (  audioObjectType == 28)
            SSCSpecificConfig();

        if (  extensionAudioObjectType != 5 &&
            bits_to_decode() >= 16 ) {
            syncExtensionType;                                          11          bslbf
            if (syncExtensionType == 0x2b7) {
                extensionAudioObjectType;                               5           uimsbf
                if ( extensionAudioObjectType == 5 ) {
                    sbrPresentFlag;                                     1           uimsbf
                    If (sbrPresentFlag == 1) {
                        extensionSamplingFrequencyIndex;                4           uimsbf
                        if ( extensionSamplingFrequencyIndex == 0xf )
                            extensionSamplingFrequency;                 24          uimsbf
                    }
                }
            }
        }
}
```

In Part 3: Audio, Subpart 1, in subclause 1.6.2.2.1 Overview, replace table 1.9 by the following table:

Table 1.9 – Audio Object Types

Audio Object Type	Object Type ID	definition of elementary stream payloads and detailed syntax	Mapping of audio payloads to access units and elementary streams
AAC MAIN	1	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
AAC LC	2	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
AAC SSR	3	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
AAC LTP	4	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.2
SBR	5	ISO/IEC 14496-3 subpart 4	
AAC scalable	6	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.3
TwinVQ	7	ISO/IEC 14496-3 subpart 4	
CELP	8	ISO/IEC 14496-3 subpart 3	
HVXC	9	ISO/IEC 14496-3 subpart 2	
TTSI	12	ISO/IEC 14496-3 subpart 6	
Main synthetic	13	ISO/IEC 14496-3 subpart 5	
Wavetable synthesis	14	ISO/IEC 14496-3 subpart 5	
General MIDI	15	ISO/IEC 14496-3 subpart 5	
Algorithmic Synthesis and Audio FX	16	ISO/IEC 14496-3 subpart 5	
ER AAC LC	17	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER AAC LTP	19	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER AAC scalable	20	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER Twin VQ	21	ISO/IEC 14496-3 subpart 4	
ER BSAC	22	ISO/IEC 14496-3 subpart 4	
ER AAC LD	23	ISO/IEC 14496-3 subpart 4	see subclause 1.6.2.2.2.1.4
ER CELP	24	ISO/IEC 14496-3 subpart 3	
ER HVXC	25	ISO/IEC 14496-3 subpart 2	
ER HILN	26	ISO/IEC 14496-3 subpart 7	
ER Parametric	27	ISO/IEC 14496-3 subpart 2 and 7	
SSC	28	ISO/IEC 14496-3 subpart 8	

Create Part 3: Audio, Subpart 8:

Subpart 8: Technical description of parametric coding for high quality audio

8.1 Scope

This part of ISO/IEC 14496 describes the MPEG-4 audio parametric coding scheme for compression of high quality audio. The short name is SSC (SinuSoidal Coding). At bit-rates around 24 kbit/s stereo and at a sampling rate of 44.1 kHz, the SSC coding scheme offers a quality that is interesting for a number of applications.

SSC employs four different tools that together parameterize an audio signal. These tools consist of transient modelling, sinusoidal modelling, noise modelling and stereo image modelling. One of the distinctive features of SSC is that it provides decoder support for independent tempo and pitch scaling at hardly any additional complexity.

Transient tool

The transient tool captures the highly dynamic events of the audio input signal. These events are efficiently modelled by means of a limited number of sinusoids that are shaped by means of an envelope.

Sinusoidal tool

The sinusoidal tool captures the deterministic events of the audio input signal. The slowly varying nature of sinusoidal components for typical audio signals is exploited by linking sinusoids over consecutive frames. By means of differential coding, the frequency, amplitude and phase parameters can be efficiently represented.

Noise tool

The noise tool captures the stochastic or non-deterministic events of the audio input signal. In the decoder, a white noise generator is used as excitation. A temporal and spectral envelope is applied to control the temporal and spectral properties of the noise in the audio signal.

Parametric stereo coding tool

The parametric stereo coding tool is able to capture the stereo image of the audio input signal into a limited number of parameters, requiring only a small overhead ranging from a few kbit/s for medium quality, up to about 9 kbit/s for higher quality. Together with a monaural downmix of the stereo input signal generated by the parametric stereo coding tool, the parametric stereo decoding tool is able to regenerate the stereo signal. It is a generic tool that in principle can operate in combination with any monaural coder. In Annex A of this document a normative description of the combination of HE-AAC with the parametric stereo coding tool is provided. SSC can also operate in dual mono mode. In that case the parametric stereo coding tool is not employed. The parametric stereo tool is intended for low bit-rates.

8.2 Terms and definitions

8.2.1
Frame
Basic unit that can be decoded on itself (file header information is required for general decoder settings).

8.2.2
Laguerre filter
Filter structure used in the noise analysis and synthesis.

8.2.3
Audio frame
Contains all data to decode an SSC-coded frame as a stand-alone unit (file header information is required for general decoder settings). For audio frames with refresh_sinusoids==%1 and refresh_noise==%1 the complete frame can always be reconstructed; otherwise it is possible in the case of random access that parts of the signal cannot be reconstructed (e.g. sinusoidal continuations, noise).

8.2.4
Sub-frame
Fine granularity within a frame.

8.2.5
f_s
The sampling frequency in Hertz.

8.2.6
Segment
An interval of samples that can be synthesized on the basis of the parameters that correspond to a sub-frame. The segment size is 2*S (see Table 8.11).

8.2.7
Window
A function that is used to weigh synthesized samples within a segment such that a valid synthesis is obtained.

8.2.8
LSF
Line Spectral Frequency.

8.2.9
Overlap and add
An additive method of combining overlapping intervals during signal synthesis.

8.2.10
Linking process
A method to keep track of sinusoidal components over time.

8.2.11
birth
The first component of a sinusoidal track.

8.2.12
Continuation
A sinusoidal track component that is not at the start or the end of a track.

8.2.13
Death
The last component of a sinusoidal track.

8.2.14
SMR
Signal-to-masking ratio.

8.2.15
Partial
Sinusoid of a limited duration.

8.2.16
IID
Inter-channel Intensity Differences.

8.2.17
IPD
Inter-channel Phase Differences.

8.2.18
OPD
Overall Phase Differences.

8.2.19
ICC
Inter-channel Coherence.

8.3 Symbols and abbreviations

8.3.1 Arithmetic operators

⌊x⌋ Round x towards minus infinity

⌈x⌉ Round x towards plus infinity.

mod Modulus operator: $\mathrm{mod}(x, y) = x - \left\lfloor \frac{x}{y} \right\rfloor y$. Defined only for positive values of *x* and *y*.

$\Gamma(\alpha)$ Gamma distribution function, defined as $\Gamma(\alpha) = \int_0^\infty e^{-t} \cdot t^{\alpha-1} dt$.

8.3.2 Relation operators

x?y:z If x is true then y else z.

8.3.3 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

uimsbf Unsigned integer, most significant bit first.

simsbf Signed integer, most significant bit first.

bslbf Bitstream left bit first.

8.3.4 Ranges

[0, 10] A number in the range of 0 up to and including 10.

[0, 10> A number in the range of 0 up to but excluding 10.

8.3.5 Number notation

%X Binary number representation (e.g. %01111100).

$X Hexadecimal number representation (e.g. $7C).

X Numbers with no prefix use decimal representation (e.g. 124).

8.3.6 Definitions

S	Number of samples in a sub-frame (see Table 8.11).
L	Number of samples in a segment; L = 2*S.
numQMFSlots	Number of QMF subband samples per ps_data() element. For SSC, this parameter is fixed to 24.

8.4 Payloads for the audio object type SSC

8.4.1 Decoder configuration (SSCSpecificConfig)

Table 8.1 – Syntax of SSCSpecificConfig()

Syntax	Num. bits	Mnemonic
SSCSpecificConfig (channelConfiguration)		
{		
decoder_level	**2**	**uimsbf**
update_rate	**4**	**uimsbf**
synthesis_method	**2**	**uimsbf**
if (channelConfiguration != 1)		
{		
mode_ext	**2**	**uimsbf**
if ((channelConfiguration == 2) && (mode_ext == 1))		
{		
reserved	**2**	**uimsbf**
}		
}		
}		

8.4.2 SSC Bitstream Payload

Table 8.2 – Syntax of ssc_audio_frame()

Syntax	Num. bits	Mnemonic
ssc_audio_frame ()		
{		
ssc_audio_frame_header()		
ssc_audio_frame_data()		
}		

Table 8.3 – Syntax of ssc_audio_frame_header()

Syntax	Num. bits	Mnemonic
ssc_audio_frame_header ()		
{		
refresh_sinusoids	**1**	**uimsbf**
refresh_sinusoids_next_frame	**1**	**uimsbf**
refresh_noise	**1**	**uimsbf**
for (ch = 0; ch < nrof_channels; ch++)		
{		
s_nrof_continuations[0][ch]	Note 1	**uimsbf**
}		
n_nrof_den	**5**	**uimsbf**
n_nrof_lsf	Note 1	**uimsbf**
freq_granularity	**2**	**uimsbf**
amp_granularity	**2**	**uimsbf**
phase_jitter_present	**1**	**uimsbf**
if (phase_jitter_present == 1)		
{		
phase_jitter_percentage	**2**	**uimsbf**
phase_jitter_band	**2**	**uimsbf**
}		
}		
Note 1: See description of s_nrof_continuations and n_nrof_lsf in section 8.5.2.		

Table 8.4 – Syntax of ssc_audio_frame_data()

Syntax	Num. bits	Mnemonic
ssc_audio_frame_data()		
{		
for (sf = 0; sf < nrof_subframes; sf++)		
{		
for (ch = 0; ch < nrof_channels; ch++)		
{		
ssc_mono_subframe(sf,ch)		
if ((channelConfiguration == 2) && (mode_ext == 1)		
&& (mod(sf+1,4)==0))		
{		
ps_data()		
}		
}		
}		
}		

Table 8.5 – Syntax of ssc_mono_subframe()

Syntax	Num. bits	Mnemonic
ssc_mono_subframe (sf,ch)		
{		
subframe_transients(sf, ch)		
subframe_sinusoids(sf, ch)		
subframe_noise(sf, ch)		
}		

Table 8.6 – Syntax of subframe_transients()

Syntax	Num. bits	Mnemonic
subframe_transients (sf, ch)		
{		
t_transient_present[sf][ch]	**1**	**uimsbf**
if (t_transient_present[sf][ch] == 1)		
{		
t_loc[sf][ch]	Note 1	**uimsbf**
if (**t_type[sf][ch]**==1)	**2**	**uimsbf**
{		
t_b_par[sf][ch]	**3**	**uimsbf**
t_chi_par[sf][ch]	**3**	**uimsbf**
t_nrof_sin[sf][ch]	**3**	**uimsbf**
t_nrof_sin[sf][ch]++		
for (i = 0; i < t_nrof_sin[sf][ch]; i++)		
{		
t_freq[sf][ch][i]	**9**	**uimsbf**
t_amp[sf][ch][i]	**5**	**uimsbf**
t_phi[sf][ch][i]	**5**	**simsbf**
}		
}		
}		
}		
Note 1: See description of t_loc in section 8.5.2.		

Table 8.7 – Syntax of subframe_sinusoids()

Syntax	Num. bits	Mnemonic
subframe_sinusoids(sf, ch)		
{		
n = 0; p=0; q=0;		
/* Continuations */		
if (sf > 0)		
{		
noc = 0;		
while (tmp_cont[ch][noc] > 0) { noc++;}		
s_nrof_continuations[sf][ch] = noc;		
}		
if ((refresh_sinusoids == 1) && (sf == 0))		
{		
for (i = 0; i < s_nrof_continuations[sf][ch]; i++, n++)		
{		
s_cont[sf][ch][n] = ssc_huff_dec(huff_scont,**bs_codeword**);	**2..5**	**bslbf**
s_freq_coarse[sf][ch][n] = ssc_huff_dec(huff_sfreqc,**bs_codeword**);	**7..25**	**bslbf**
s_freq_fine[sf][ch][n]	**0..3**	**simsbf**
s_amp_coarse[sf][ch][n] = ssc_huff_dec(huff_sampca,**bs_codeword**);	**3..16**	**bslbf**
s_amp_fine[sf][ch][n]	**0..3**	**simsbf**
s_phi[sf][ch][n]	**5**	**simsbf**
if (s_cont[sf][ch][n] > 0)		
{		
s_adpcm_grid[sf][ch][n] = ssc_huff_dec(huff_sgrid,**bs_codeword**);	**3..7**	**bslbf**
s_delta_cont_freq_pha[sf+1][ch][p]	**2**	**uimsbf**
p++;		
}		
if (s_cont[sf][ch][n] > 1)		
{		
s_delta_cont_freq_pha[sf+2][ch][q]	**2**	**uimsbf**
q++;		
}		
}		
}		
else {		
for (i = 0; i < s_nrof_continuations[sf][ch]; i++, n++)		
{		
if (sf == 0)		
{		
s_cont[sf][ch][n] = ssc_huff_dec(huff_scont,**bs_codeword**);	**2..5**	**bslbf**
}		
else {		
s_cont[sf][ch][n] = tmp_cont[ch][n] - 1;		
}		
if (s_cont[sf][ch][n] > 0)		
{		
p++;		
}		
if (s_cont[sf][ch][n] > 1)		
{		
if ((refresh_sinusoids_next_frame == 0) || (nrof_subframes-sf > 2))		
{		
s_delta_cont_freq_pha[sf+2][ch][q]	**2**	**uimsbf**
}		
q++;		
}		
s_delta_cont_amp[sf][ch][n] =	**1..15**	**bslbf**
ssc_huff_dec(huff_sampcr[amp_granularity],**bs_codeword**);		

}
}
/* *Births* */
s_nrof_births[sf][ch] = ssc_huff_dec(huff_nrofbirths,**bs_codeword**);
	3..15	**bslbf**

if (s_nrof_births[sf][ch] > 0)
{
s_cont[sf][ch][n] = ssc_huff_dec(huff_scont,**bs_codeword**);
	2..5	**bslbf**

s_freq_coarse[sf][ch][n] = ssc_huff_dec(huff_sfreqba,**bs_codeword**);	**7..21**	**bslbf**
s_freq_fine[sf][ch][n]	**0..3**	**simsbf**
s_amp_coarse[sf][ch][n] = ssc_huff_dec(huff_sampba,**bs_codeword**);	**3..15**	**bslbf**
s_amp_fine[sf][ch][n]	**0..3**	**simsbf**

s_phi[sf][ch][n]
if (s_cont[sf][ch][n] > 0)
{
if ((refresh_sinusoids_next_frame == 0) || (nrof_subframes-sf > 1))
{
	5	**simsbf**

s_delta_cont_freq_pha[sf+1][ch][p]	**2**	**uimsbf**

}
p++;
}
if (s_cont[sf][ch][n] > 1)
{
if ((refresh_sinusoids_next_frame == 0) || (nrof_subframes-sf > 2))
{		

s_delta_cont_freq_pha[sf+2][ch][q]	**2**	**uimsbf**

}
q++;
}
n++;
for (i = 1; i < s_nrof_births[sf][ch]; i++, n++)
{
s_cont[sf][ch][n] = ssc_huff_dec(huff_scont,**bs_codeword**);
	2..5	**bslbf**

s_delta_birth_freq_coarse[sf][ch][n] = ssc_huff_dec(huff_sfreqbr,**bs_codeword**);	**5..23**	**bslbf**
s_delta_birth_freq_fine[sf][ch][n]	**0..3**	**simsbf**
s_delta_birth_amp_coarse[sf][ch][n] = ssc_huff_dec(huff_sampbr,**bs_codeword**);	**2..21**	**bslbf**
s_delta_birth_amp_fine[sf][ch][n]	**0..3**	**simsbf**

s_phi[sf][ch][n]
if (s_cont[sf][ch][n] > 0)
{
if ((refresh_sinusoids_next_frame == 0) || (nrof_subframes-sf > 1))
{
	5	**simsbf**

s_delta_cont_freq_pha[sf+1][ch][p]	**2**	**uimsbf**

}
p++;
}
if (s_cont[sf][ch][n] > 1)
{
if ((refresh_sinusoids_next_frame == 0) || (nrof_subframes-sf > 2))
{		

s_delta_cont_freq_pha[sf+2][ch][q]	**2**	**uimsbf**

}
q++;
}
}
}
/* Keep track of sinusoids that continue in next sub-frame(s) */
for (i = 0, k = 0; i < n; i++)
{		

```
            if (s_cont[sf][ch][i] > 0)
            {
                tmp_cont[ch][k] = s_cont[sf][ch][i];
                k++;
            }
        }
}
```

Note: The variables p, q are used as position indices for subframe+1 and subframe+2 respectively.

Table 8.8 – Syntax of subframe_noise()

Syntax	Num. bits	Mnemonic
subframe_noise (sf, ch)		
{		
if ((refresh_noise == 1) && (sf == 0))		
{		
n_laguerre[ch]	**2**	**uimsbf**
n_laguerre_granularity[sf][ch]	**1**	**uimsbf**
for (i = 0; i < n_nrof_den; I++)		
{		
n_lar_den_coarse[sf][ch][i] = ssc_huff_dec(huff_nlag,**bs_codeword**);	**1..18**	**bslbf**
If (n_laguerre_granularity[sf][ch]==1)		
{		
n_lar_den_fine[sf][ch][i]	**2**	**simsbf**
}		
}		
n_gain[sf][ch]	**7**	**uimsbf**
n_lsf[sf][ch][0] = ssc_huff_dec(huff_nlsf,**bs_codeword**);	**2..9**	**bslbf**
for (i = 1; i < n_nrof_lsf; i++)		
{		
n_delta_lsf[sf][ch][i] = ssc_huff_dec(huff_nlsf,**bs_codeword**);	**2..9**	**bslbf**
}		
}		
else {		
if (mod(sf,2) == 0)		
{		
n_laguerre_granularity[sf][ch]	**1**	**uimsbf**
for (i = 0; i < n_nrof_den; i++)		
{		
n_delta_lar_den_coarse[sf][ch][i] = ssc_huff_dec(huff_nlag,**bs_codeword**);	**1..18**	**bslbf**
if(n_laguerre_granularity[sf][ch]==1)		
{		
n_delta_lar_den_fine[sf][ch][i]	**2**	**simsbf**
}		
}		
}		
if (mod(sf,4) == 0)		
{		
n_delta_gain[sf][ch] = ssc_huff_dec(huff_ngain,**bs_codeword**);	**1..12**	**bslbf**
if (**n_overlap_lsf** == 1)	**1**	**uimsbf**
{		
for (i = n_nrof_overlap_lsf; i < n_nrof_lsf; i++)		
{		
n_delta_lsf[sf][ch][i] = ssc_huff_dec(huff_nlsf,**bs_codeword**);	**2..9**	**bslbf**
}		
}		
else {		
n_lsf[sf][ch][0] = ssc_huff_dec(huff_nlsf,**bs_codeword**);	**2..9**	**bslbf**
for (i = 1; i < n_nr_of_lsf; i++)		
{		
n_delta_lsf[sf][ch][i] = ssc_huff_dec(huff_nlsf,**bs_codeword**);	**2..9**	**bslbf**
}		
}		
}		
}		
}		

Table 8.9 – Syntax of ps_data()

Syntax	Num. bits	Mnemonic
ps_data()		
{		
if (**enable_ps_header**) {	**1**	**uimsbf**
if (**enable_iid**) {	**1**	**uimsbf**
iid_mode	**3**	**uimsbf**
nr_iid_par = nr_iid_par_tab[iid_mode]		
nr_ipdopd_par = nr_ipdopd_par_tab[iid_mode]		
}		
if (**enable_icc**) {	**1**	**uimsbf**
icc_mode	**3**	**uimsbf**
nr_icc_par = nr_icc_par_tab[icc_mode]		
}		
enable_ext	**1**	**uimsbf**
}		
frame_class	**1**	**uimsbf**
num_env_idx	**2**	**uimsbf**
num_env = num_env_tab[frame_class][num_env_idx]		
if (frame_class) {		
for (e=0 ; e<num_env ; e++) {		
border_position[e]	**5**	**uimsbf**
}		
}		
for (e=0 ; e<num_env ; e++) {		
if (enable_iid) {		
iid_dt[e]	**1**	**uimsbf**
iid_data()		
}		
}		
for (e=0 ; e<num_env ; e++) {		
if (enable_icc) {		
icc_dt[e]	**1**	**uimsbf**
icc_data()		
}		
}		
if (enable_ext) {		
cnt = **ps_extension_size**	**4**	**uimsbf**
if (cnt == 15)		
cnt += **esc_count**	**8**	**uimsbf**
num_bits_left = 8 * cnt		
while (num_bits_left > 7) {		
ps_extension_id	**2**	**uimsbf**
num_bits_left -= 2		
ps_extension(ps_extension_id, num_bits_left)		
}		
fill_bits	num_bits_left	
}		
}		
ps_extension(ps_extension_id, num_bits_left){		
if (ps_extension_id == 0) {		
if (**enable_ipdopd**) {	**1**	**uimsbf**

for (e=0 ; e<num_env ; e++) {		
ipd_dt[e]	**1**	**uimsbf**
ipd_data()		
opd_dt[e]	**1**	**uimsbf**
opd_data()		
num_bits_left -= ipd_bits + opd_bits + 2		Note 1
}		
}		
reserved_ps	**1**	**uimsbf**
num_bits_left -= 2		
}		
}		
iid_data() {		
if (iid_dt[e]) {		
for (b=0 ; b<nr_iid_par; b++) {		
iid_par_dt[e][b] = ssc_huff_dec(huff_iid_dt[iid_quant]**,bs_codeword**);	**1…20**	Note 2
}		
}		
else {		
for (b=0 ; b<nr_iid_par; b++) {		
iid_par_df[e][b] = ssc_huff_dec(huff_iid_df[iid_quant]**,bs_codeword**);	**1…18**	Note 2
}		
}		
}		
icc_data() {		
if (icc_dt[e]) {		
for (b=0 ; b<nr_icc_par; b++) {		
icc_par_dt[e][b] = ssc_huff_dec(huff_icc_dt**,bs_codeword**);	**1…14**	**bslbf**
}		
}		
else {		
for (b=0 ; b<nr_icc_par; b++) {		
icc_par_df[e][b] = ssc_huff_dec(huff_icc_df**,bs_codeword**);	**1…13**	**bslbf**
}		
}		
}		
ipd_data() {		
if (ipd_dt[e]) {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
ipd_par_dt[e][b] = ssc_huff_dec(huff_ipd_dt**,bs_codeword**);	**1…5**	**bslbf**
}		
}		
else {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
ipd_par_df[e][b] = ssc_huff_dec(huff_ipd_df**,bs_codeword**);	**1…4**	**bslbf**
}		
}		
}		
opd_data() {		
if (opd_dt[e]) {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
opd_par_dt[e][b] = ssc_huff_dec(huff_opd_dt**,bs_codeword**);	**1…5**	**bslbf**
}		
}		
else {		
for (b=0 ; b<nr_ipdopd_par; b++) {		

opd_par_df[e][b] = ssc_huff_dec(huff_opd_df,**bs_codeword**);
}
}
}
	1…5	**bslbf**

Note 1: ipd_bits and opd_bits represent the number of bits read by ipd_data() and opd_data() respectively. Note 2: the index iid_quant into huff_iid_df is obtained from Table 8.19.		

8.5 Semantics

8.5.1 SSCSpecificConfig

channelConfiguration – Channel configuration as defined in ISO/IEC-14496-3:edition 2001 subpart 1, paragraph 1.6.3.4.

decoder_level – Complexity bounds for decoder settings. A decoder that supports a certain level of complexity is not able to decode a bit-stream that is encoded according a higher level of complexity. This decoder is however able to decode a bit-stream that is encoded according to a lower level of complexity (see Table 8.10).

Table 8.10 - Decoder level

decoder_ level	Level of complexity	max_nrof_sinusoids	max_nrof_den	#bits for s_nrof_ continuations	#bits for n_nrof_lsf
00	Reserved	Na	Na	Na	Na
01	Medium	60	22	6	4
10	Reserved	Na	Na	Na	Na
11	Reserved	Na	Na	Na	Na

max_nrof_sinusoids - Maximum number of sinusoids that is allowed (sinusoids under Meixner transients not included).

max_nrof_den - Maximum value for n_nrof_den.

update_rate – Four bits indicating the sub-frame size S. Table 8.11 shows the relationship between update_rate and the sub-frame size S in samples.

Table 8.11 - Update rate

update_rate	S	update_rate	S
0000	Reserved	1000	Reserved
0001	Reserved	1001	Reserved
0010	Reserved	1010	Reserved
0011	Reserved	1011	Reserved
0100	384	1100	Reserved
0101	Reserved	1101	Reserved
0110	Reserved	1110	Reserved
0111	Reserved	1111	Reserved

synthesis_method – Two bits providing information on the preferred synthesis for the specific encoded program (see Table 8.12).

Table 8.12 - Synthesis method

Synthesis_method	Optimal synthesis
00	Overlap and Add
01	Reserved
10	Reserved
11	Reserved

mode_ext – In combination with channelConfiguration the mode_ext bits provide the full channel configuration. The number of bits is dependent on the channelConfiguration (see Table 8.13).

Table 8.13 - Channel configuration

channelConfiguration	# bits for mode_ext	nrof_channels
1	0	1
2	2	According to mode_ext
0, 3 … 15	Na	Na

For channelConfiguration == 2, Table 8.14 applies:

Table 8.14 – Channel configuration in case channelConfiguration == 2

mode_ext	Full channel configuration
00	Dual mono (ch0=left, ch1=right)
01	Parametric Stereo
10	Reserved
11	Reserved

reserved – Two reserved bits; should be set to %0.

8.5.2 Decoding of SSC Bitstream Payload

ssc_audio_frame() – syntactic element that contains a single SSC frame

ssc_audio_frame_header() – syntactic element that contains the header data for a single SSC frame

ssc_audio_frame_data() – syntactic element that contains the data for a single SSC frame

ssc_huff_dec() – Huffman decoding procedure. See Annex B.

refresh_sinusoids – One bit indicating how sinusoidal continuations of the first sub-frame in a frame are encoded. If this bit equals %0, the continued track data is differentially coded with respect to the last sub-frame of the previous frame. If this bit equals %1, the continued track data in the first sub-frame of a frame are coded as absolute values.

refresh_sinusoids_next_frame – One bit providing an additional frame look ahead for the ADPCM decoding of sinusoidal parameters. If this bit is set to %1, the next frame is a refresh frame. In that case the bit refresh_sinusoids shall be set to %1 in the next frame.

refresh_noise – One bit indicating how noise parameters of the first sub-frame in a frame are encoded. If this bit equals %0, the noise parameters are differentially coded with respect to the last sub-frame of the previous frame. If this bit equals %1, the noise parameters in the first sub-frame of a frame are coded as absolute values.

s_nrof_continuations[sf][ch] – For sub-frame sf and channel ch, this value represents the number of continuations. In the case sf==0 the value of s_nrof_continuations is provided in the bit-stream. For the

remaining values of sf, the value of s_nrof_continuations is obtained implicitly as described in section 8.6.2.1. The number of bits required for s_nrof_continuations[0][ch] depends on the maximum number of allowed sinusoids, which is dependent of the decoder complexity, indicated by decoder_level. This relation is shown in Table 8.10.

n_nrof_den – Number of denominator LAR coefficients of the FIR filter for noise generation.

n_nrof_lsf - Number of LSF coefficients used for the generation of envelope for noise generation. The number of bits required for n_nrof_lsf depends on the decoder complexity, indicated by decoder_level. This relation is shown in Table 8.10.

freq_granularity – The granularity of the differentially or absolute coded frequency parameters used in subframe_sinusoids(). This parameter determines the number of bits to be read for the fine part of the frequency parameters.

amp_granularity – The granularity of the differentially or absolute coded amplitude parameters used in subframe_sinusoids(). This parameter determines the Huffman table to be used or the number of bits to be read for the fine part of the amplitude parameters.

phase_jitter_present – One bit to indicate presence of phase jitter parameters. If this bit equals %0, no phase jitter is present. If this bit equals %1, phase jitter is present.

phase_jitter_percentage – This is a two bit unsigned integer identifying a distance percentage. The full distance equals half a quantisation step. The maximum jitter applied to the frequency components is

$$max\_jitter = 2^{\text{freq\_granularity}-1}\,\frac{\text{phase\_jitter\_percentage}+1}{2^2}.$$

phase_jitter_band – Two bits identifying the frequency representation level from which phase jitter must be applied. Table 8.15 provides the relation between phase_jitter_band and $f_{jitter,min}$.

Table 8.15 - Phase jitter band expressed in representation levels

phase_jitter_band	frequency representation level $f_{jitter,min}$
00	0
01	800
10	1600
11	2400

nrof_subframes – the number of sub-frames in one frame. This value is fixed to 8.

ssc_mono_subframe() – syntactic element that contains the data for a single SSC sub frame.

ps_data() – syntactic element that contains the parametric stereo data.

subframe_transients() – syntactic element that contains the transient data for a single SSC sub frame.

subframe_sinusoids() – syntactic element that contains the sinusoid data for a single SSC sub frame.

subframe_noise() – syntactic element that contains the noise data for a single SSC sub frame.

t_transient_present[sf][ch] – One bit indicating if a transient is present in sub-frame sf, channel ch. If t_transient_present[sf][ch]==%1, a transient is present. If t_transient_present[sf][ch]==%0, no transient is present.

t_loc[sf][ch] – Indication of the location of the transient in sub-frame sf of channel ch, indicated in the number of samples from the start of the sub-frame. The valid range for t_loc is [0,S>. The number of bits that is used to represent t_loc is calculated according to

$$\lceil \log_2(S) \rceil,$$

where S represents the sub-frame size in samples.

t_type[sf][ch] – Two bits to indicate the transient type of the transient in sub-frame sf of channel ch (see Table 8.16).

Table 8.16 - Transient types

t_type	Type
00	Step
01	Meixner
10	Reserved
11	Reserved

t_b_par[sf][ch] – For a transient of the Meixner type in sub-frame sf of channel ch, these 3 bits hold the value for the attack of the transient envelope, denoted as the '*b*-parameter'. Allowed values for t_b_par are [0, 1, 2, 3]. The remaining values are reserved. The value *b* is calculated as

b = t_b_par + 2.

t_chi_par[sf][ch] – For a transient of the Meixner type in sub-frame sf of channel ch, these 3 bits hold the value for the decay of the transient envelope, denoted as the 'ξ-parameter'. Allowed values for t_chi_par are [0, 1, 2, 3]. The remaining values are reserved. The value ξ is tabulated in Table 8.17.

Table 8.17 - Quantised values for b and ξ

ξ		t_b_par			
		0	1	2	3
t_chi_par	0	0.9688	0.9685	0.9683	0.9681
	1	0.9763	0.9756	0.9750	0.9744
	2	0.9839	0.9827	0.9817	0.9807
	3	0.9914	0.9898	0.9884	0.9870

t_nrof_sin[sf][ch] – For a transient of the Meixner type in sub-frame sf of channel ch, these 3 bits represent the number of sinusoids that are present under the envelope. The number of sinusoids under the Meixner envelope is equal to the value in the stream plus one.

t_freq[sf][ch][i] – For a transient of the Meixner type in sub-frame sf of channel ch, these bits represent the frequency in radians of the i-th sinusoid under the transient envelope.

$$tf_q[i] = \frac{2\pi}{f_s} \frac{10^{\frac{\text{t\_freq[sf][ch][i]}}{11.4 \cdot 21.4}} - 1}{0.00437},$$

where tf_q represents the dequantized absolute frequency in radians.

t_amp[sf][ch][i] – For a transient of the Meixner type in sub-frame sf of channel ch, these bits represent the amplitude of the i-th sinusoid under the transient envelope.

$$ta_q[i] = ta_b^{2\cdot \text{t\_amp[sf][ch][i]}},$$

where ta_b represents the log quantisation base (maximum error=1.5dB), $ta_b = 1.1885$. ta_q represents the dequantized absolute amplitude.

t_phi[sf][ch][i] – For a transient of the Meixner type in sub-frame sf of channel ch, these bits represent the phase of the i-th sinusoid under the transient envelope. The decoded value is converted into a phase value in radians in the range [-π, π> and is specified for the start of the transient.

$$tp_q[i] = 2 \cdot tp_e \cdot \text{t\_phi[sf][ch][i]},$$

where tp_e represents the absolute phase error ($tp_e = \frac{\pi}{32}$) and tp_q represents the dequantized absolute phase (in radians). The allowed range for t_phi is [-16, 15]; the representation level +16 is represented by –16 (because +π==-π).

noc – Local variable that counts the number of continuations in the previous sub-frame.

tmp_cont[ch][noc] – Local array that contains a copy of the s_cont-parameters of the previous sub-frame, needed for parsing the stream correctly (extract number of continuations and keep track of how many sub-frames the sinusoidal track must be continued in the current frame).

s_cont[sf][ch][n] – For sub-frame sf and channel ch, this value indicates how many sub-frames component n will be continued in the current frame (if the component continues also in the next frame, one must be added to the number of sub-frames it continues in the current frame. If the value is 0 this indicates component n stops at sub-frame sf, which is called a death). The valid range for s_cont is [0, 9].

s_freq_coarse[sf][ch][n] – For sub-frame sf and channel ch, this value represents the coarse frequency parameter of the n-th sinusoid.

s_freq_fine[sf][ch][n] – For sub-frame sf and channel ch, this signed integer represents a higher level of detail to the coarse frequency parameter. The number of bits to be read amounts to (3 – freq_granularity). The frequency representation level f_{rl} is the sum of coarse frequency, fine frequency scaled to the granularity grid.

$$f_{rl}[n] = \text{s\_freq\_coarse[sf][ch][n]} + \text{s\_freq\_fine[sf][ch][n]} \cdot 2^{\text{freq\_granularity}}.$$

Phase jitter is only applied in combination with tempo and pitch scaling. If phase_jitter_present == %1 and $f_{rl} > f_{jitter,min}$, the phase jitter parameter is

$$f_{jitter} = \lfloor max\_jitter \cdot (2x - 1) + 0.5 \rfloor,$$

where x holds a random number, uniformly distributed between 0 and 1, generated for each frequency parameter in the sub-frame, matching the requirement above. The decoded value is converted into a dequantized absolute frequency value f_q in radians, using the following equation:

$$f_q[n] = \frac{2\pi}{f_s} \frac{10^{\frac{f_{rl}[n]}{91.2 \cdot 21.4}} - 1}{0.00437}.$$

s_amp_coarse[sf][ch][n] – For sub-frame sf and channel ch, this value represents the coarse amplitude parameter of the n-th sinusoid.

s_amp_fine[sf][ch][n] – For sub-frame sf and channel ch, this parameter represents a higher level of detail to the coarse amplitude parameter. The number of bits to be read amounts to (3 – amp_granularity). The amplitude representation level sa_{rl} is the sum of coarse amplitude, fine amplitude scaled to the granularity grid

$$sa_{rl}[n] = \text{s\_amp\_coarse}[sf][ch][n] + \text{s\_amp\_fine}[sf][ch][n] \cdot 2^{\text{amp\_granularity}}.$$

The decoded value is converted into a dequantized linear amplitude value sa_q in the range [1, 2^{15}-1] conforming to

$$sa_q[n] = sa_b^{2 \cdot sa_{rl}[n]}.$$

Where $sa_b = 1.0218$ is the log quantization base. Its value corresponds to a maximum error of 0.1875 dB.

s_phi[sf][ch][n] – For sub-frame sf and channel ch, this represents the phase parameter of the n-th sinusoid. This value is converted into a phase value in radians in the range [-π, π> conforming to

$$sp_q[n] = 2 \cdot sp_e \cdot \text{s\_phi}[sf][ch][n],$$

where sp_e represents the absolute phase error ($sp_e = \frac{\pi}{32}$) and sp_q represents the dequantized absolute phase (in radians) The allowed range for s_phi is [-16, 15]; the representation level +16 is represented by –16 (because +π == -π).

s_adpcm_grid[sf][ch][n] – For sub-frame sf and channel ch, this value represents the initial index into Table 8.30 as used in the ADPCM decoder for the n-th sinusoid. This table is used to decode the sinusoidal information.

s_delta_cont_freq_pha[sf][ch][n] – For sub-frame sf and channel ch, this value represent the representation levels for the n-th sinusoid, that serve as input to the ADPCM decoder. In order to compensate for the 2 sub-frames delay in the decoder, the representation levels are transmitted 2 sub-frames in advance. In the bit-stream syntax, future representation levels are indicated by indices sf+1 and sf+2, indicating the representation levels of the two sub-sequent sub-frames respectively. In the case sf+1 or sf+2 exceeds nrof_subframes, then the representation level is assigned to the next frame. In this case, the new number of the sub-frame in the next frame is (sf+1) - nrof_subframes or (sf+2) - nrof_subframes respectively.

s_delta_cont_amp[sf][ch][n] – For sub-frame sf and channel ch, this represents the differential amplitude parameter of the n-th sinusoid. This value is converted into a linear amplitude value in the range [1, 2^{15}-1] conforming to

$$sa_{rl}[n] = sa_{rl,psf} + \text{s\_delta\_cont\_amp}[sf][ch][n],$$

where sa_{rl} represents the amplitude representation level and $sa_{rl,psf}$ represents the amplitude representation level in the previous sub-frame. For dequantization of sa_{rl} into sa_q see s_amp_fine. In the case the amplitude granularity, amp_granularity of the current frame is different from that of the previous frame, prior to applying the differential encoded values, the granularity of the value of the previous frame is converted to the granularity of the current frame according to

$$sa_{rl,psf} = 2^{\text{amp\_granularity}} \left\lfloor \frac{sa'_{rl,psf}}{2^{\text{amp\_granularity}}} + 0.5 \right\rfloor,$$

where $sa'_{rl,psf}$ represents the amplitude representation level of the previous sub-frame and amp_granularity represents the granularity of the current sub-frame.

s_nrof_births[sf][ch] – For sub-frame sf and channel ch, this value represents the number of births. The allowed range is [0, max_nrof_sinusoids-s_nrof_continuations[sf][ch]].

s_delta_birth_freq_coarse[sf][ch][n] – For sub-frame sf and channel ch, this value represents the differential, coarse frequency parameter of the n-th sinusoid.

s_delta_birth_freq_fine[sf][ch][n] – For sub-frame sf and channel ch, this represents a higher level of detail to the coarse differential frequency parameter. The number of bits to be read is (3 – freq_granularity). The delta frequency representation level df_{rl} is

$$df_{rl}[n] = \text{s\_delta\_birth\_freq\_coarse}[sf][ch][n] + \text{s\_delta\_birth\_freq\_fine}[sf][ch][n] \cdot 2^{\text{freq\_granularity}}$$

The decoded value of the n-th sinusoid is converted into a frequency value in Hertz, using the frequency representation level of the previous birth f_{rl} of sub-frame sf (the (n-1)th sinusoid):

$$f_{rl}[n] = f_{rl}[n-1] + df_{rl}[n],$$

where f_{rl} represents the frequency representation level. Modification of f_{rl} due to phase jitter uses the same rules as stated under s_freq_fine. For dequantization of f_{rl} into f_q see also s_freq_fine.

s_delta_birth_amp_coarse[sf][ch][n] – For sub-frame sf and channel ch, this represents the differential, coarse amplitude parameter of the n-th sinusoid.

s_delta_birth_amp_fine[sf][ch][n] – For sub-frame sf and channel ch, this represents a higher level of detail to the coarse amplitude parameter. The number of bits to be read is (3 – amp_granularity). The delta amplitude representation level sda_{rl} is

$$sda_{rl}[n] = \text{s\_delta\_birth\_amp\_coarse}[sf][ch][n] + \text{s\_delta\_birth\_amp\_fine}[sf][ch][n] \cdot 2^{\text{amp\_granularity}}$$

The decoded value for the n-th sinusoid is converted into a linear amplitude value, using the amplitude representation level of the previous birth sa_{rl} (the (n-1)th sinusoid):

$$sa_{rl}[n] = sa_{rl}[n-1] + sda_{rl}[n],$$

where sa_{rl} represents the amplitude representation level. For dequantization of sa_{rl} into sa_q see s_amp_fine.

n_laguerre[ch] – λ coefficient of the Laguerre filter for noise synthesis, see Table 8.18.

Table 8.18 - Possible values for λ

n_laguerre	λ
00	0
01	0.5
10	0.7
11	Reserved

n_laguerre_granularity[sf][ch] – 1 bit denoting quantisation accuracy of the Laguerre coefficients.

n_lar_den_coarse[sf][ch][i] – for sub-frame sf and channel ch this represents the denominator LAR coefficient number i.

n_lar_den_fine[sf][ch][i] – for sub-frame sf and channel ch this represents a higher level of detail to the coarse denominator LAR coefficient parameter. The representation level $nlar_{rl}$ is the sum of coarse denominator LAR and fine denominator LAR:

$nlar_{rl}[i]$ = n_lar_den_coarse[sf][ch][i] + n_lar_den_fine[sf][ch][i]

and is converted to a LAR coefficient according to:

$nlar_q[i] = nlar_{rl}[i] * \Delta_{LAR}$.

For the definition of Δ_{LAR} see section 8.6.3.3.1. Section 8.6.3.3.2 and 8.6.3.3.3 show how to convert LARs to reflection- or a-parameters.

n_gain[sf][ch] – For sub-frame sf and channel ch this value represents the gain coefficient. The gain representation level $ngain_{rl}$ is obtained as:

$ngain_{rl}$ = n_gain[sf][ch]

n_lsf[sf][ch][i] – For sub-frame sf and channel ch this value represents the LSF coefficient number i. The allowed range for n_lsf is [0,255]. The dequantised LSF parameters $nlsf_q$ are obtained as

$nlsf_q[i]$ = n_lsf[sf][ch][i] * π / 256 + π / 512

n_delta_lsf[sf][ch][i] – For sub-frame sf and channel ch this value represents the differential LSF coefficient number i. They are obtained using the following algorithm

for $i > 0$:

$nlsf_q[i] = (nlsf_q[i\text{-}1]$ + n_delta_lsf[sf][ch][i]) * π / 256 + π / 512.

n_delta_lar_den_coarse[sf][ch][i] – for sub-frame sf and channel ch this represents the differential, denominator LAR coefficient number i.

n_delta_lar_den_fine[sf][ch][i] – for sub-frame sf and channel ch this represents a higher level of detail to the coarse denominator LAR coefficient parameter. The representation level $ndlar_{rl}$ is the sum of the differential coarse denominator LAR and differential fine denominator LAR:

$ndlar_{rl}[i]$ = n_delta_lar_den_coarse[sf][ch][i] + n_delta_lar_den_fine[sf][ch][i]

and is converted to a LAR coefficient according to:

$nlar_{rl}[i] = nlar_{rl,psf}[i] + ndlar_{rl}[i]$.

where $nlar_{rl}[i]$ and $nlar_{rl,psf}[i]$ represent the LAR representation level of the current and previous sub-frame respectively. Note that in the case the n_laguerre_granularity changes from %1 to %0 going from sub-frame sf-1 to sf, the value $nlar_{rl,psf}[i]$ is first converted to the coarsest quantisation grid according to:

$nlar_{rl,psf}[i] = 4 * \lfloor nlar'_{rl,psf}[i] / 4 + 0.5 \rfloor$.

where $nlar'_{rl,psf}[i]$ represents the LAR representation level of the previous sub-frame. For further processing of $nlar_{rl}$ into reflection or a-parameters see sections 8.6.3.3.2 and 8.6.3.3.3.

n_delta_gain[sf][ch] – For sub-frame sf and channel ch this value represents the differential gain coefficient, and is converted to a representation level $ngain_{rl}$ according to:

$ngain_{rl} = ngain_{rl,p4sf}$ + n_delta_gain[sf][ch],

where $ngain_{rl,p4sf}$ represents the gain representation level for sub-frame sf-4.

n_overlap_lsf – One bit indicating whether LSF coefficients overlap from the previous definition in channel ch.

enable_ps_header – One bit indicating whether PS header information is present. If set to %1, the PS header data configuring the PS decoder is transmitted. Otherwise, the latest configuration persists.

enable_iid – One bit denoting the presence of IID parameters. If enable_iid is set to %1, Inter-channel Intensity Difference (IID) parameters will be sent from this point on in the bit stream. If enable_iid==%0, no IID parameters will be sent from this point on in the bit stream.

iid_mode - The configuration of IID parameters (number of bands and quantisation grid, iid_quant) is determined by iid_mode. Eight different configurations for IID parameters are supported (see Table 8.19).

Table 8.19 - IID mode configurations

iid_mode	nr_iid_par_tab	nr_ipdopd_par_tab	iid_quant	Index range
0 (000)	10	5	0	-7…7
1 (001)	20	11		-7…7
2 (010)	34	17		-7…7
3 (011)	10	5	1	-15…15
4 (100)	20	11		-15…15
5 (101)	34	17		-15…15
6 (110)	reserved			
7 (111)	reserved			

If no IID data is sent in the bit-stream, all IID parameters are reset to 0 (i.e. index=0).

The default and the fine quantization grids for IID, iid_quant = %0 and iid_quant = %1 are as provided in Table 8.B.18 and Table 8.B.17 respectively.

Table 8.20 - Default quantization grid for IID.

Index	-7	-6	-5	-4	-3	-2	-1	0
IID [dB]	-25	-18	-14	-10	-7	-4	-2	0
Index	1	2	3	4	5	6	7	
IID [dB]	2	4	7	10	14	18	25	

Table 8.21 - Fine quantization grid for IID.

Index	-15	-14	-13	-12	-11	-10	-9	-8
IID [dB]	-50	-45	-40	-35	-30	-25	-22	-19
Index	-7	-6	-5	-4	-3	-2	-1	0
IID [dB]	-16	-13	-10	-8	-6	-4	-2	0
Index	1	2	3	4	5	6	7	8
IID [dB]	2	4	6	8	10	13	16	19
Index	9	10	11	12	13	14	15	
IID [dB]	22	25	30	35	40	45	50	

Note that the configuration of the Inter-channel Phase Difference (IPD) / Overall Phase Difference (OPD) parameters, is strictly coupled to the IID configuration. This is also illustrated in Table 8.19.

enable_icc – One bit denoting the presence of ICC parameters. If enable_icc is set to %1, Inter-channel Coherence (ICC) parameters will be sent from this point on in the bit stream. If enable_icc==%0, no ICC parameters will be sent from this point on in the bit stream.

icc_mode – The configuration of Inter-channel Coherence parameters (number of bands and quantisation grid) is determined by icc_mode. Eight different configurations are supported for IC parameters (see Table 8.22).

Table 8.22 - ICC mode configuration

icc_mode	nr_icc_par_tab	Index range	Mixing procedure
0 (000)	10	0…7	R_a
1 (001)	20	0…7	
2 (010)	34	0…7	
3 (011)	10	0…7	R_b
4 (100)	20	0…7	
5 (101)	34	0…7	
6 (110)	reserved		
7 (111)	reserved		

If no ICC data is sent in the bit-stream, all ICC parameters are reset to 1 (i.e. index=0). The default quantization grid for ICC is provided in Table 8.B.19.

Table 8.23 - Quantization grid for ICC.

index	0	1	2	3	4	5	6	7
ρ	1	0.937	0.84118	0.60092	0.36764	0	-0.589	-1

enable_ext – The PS extension layer is enabled using the enable_ext bit. If it is set to %1 the IPD and OPD parameters are sent. If it's disabled, i.e. %0, the extension layer is skipped.

frame_class – The frame_class bit determines whether the parameter positions of the current frame are uniformly spaced accross the frame (FIX_BORDERS: frame_class==%0) or they are defined using the positions described by border_position (VAR_BORDERS: frame_class==%1).

num_env_idx – The number of (sets of) parameters (envelopes) per frame is determined using num_env_idx. In case of fixed parameter spacing (frame_class==%0) and a variable parameter spacing (frame_class==%1) this relation is shown in Table 8.24.

num_env – Local variable denoting the number of stereo envelopes (sets of parameters). num_env==0 signals that no new stereo parameters are transmitted and that the last parameters in the previous ps_data() element shall be kept unchanged and applied to the current ps_data() element.

Table 8.24 - Number of parameter sets num_env as a function of num_env_idx in case of fixed and variable spacing.

	num_env_tab[frame_class][num_env_idx]	
num_env_idx	frame_class==0	frame_class==1
0	0	1
1	1	2
2	2	3
3	4	4

border_position[e] – In case of variable parameter spacing the parameter positions are determined by border_position[e]. It contains the QMF sample index n_e for parameter set e of the current ps_data() element.

iid_dt[e] – This flag describes for envelope index e, whether the IID parameters are coded differentially over time (iid_dt==%1) or over frequency (iid_dt==%0). In the case iid_mode is different from the previous envelope (e-1), iid_dt[e] shall have the value 0% forcing frequency differential coding.

iid_data() – syntactic element containing IID data.

icc_dt[e] – This flag describes for envelope index e, whether the ICC parameters are coded differentially over time (icc_dt==%1) or over frequency (icc_dt==%0). In the case icc_mode is different from the previous envelope (e-1), icc_dt[e] shall have the value 0% forcing frequency differential coding.

icc_data() – syntactic element containing ICC data.

cnt – Local variable denoting the number of bytes used for the ps_extension() element.

ps_extension_size – The length of the PS extension layer is ps_extension_size, measured in bytes. If the extension size makes use of the escape code (ps_extension_size == 15) the length of the extension layer is extended by an additional amount of bytes.

esc_count – In case the escape code (ps_extension_size == 15) is used, esc_count describes the additional length of the PS extension layer measured in bytes.

num_bits_left – Local variable describing the number of bits left for reading in the ps_extension() element.

ps_extension_id – The identification tag (version) of the PS extension layer is given by ps_extension_id. Currently only one version is supported (see Table 8.25).

Table 8.25 - Description of ps_extension_id

ps_extension_id	Version
00 (0)	v0
01 (1)	reserved
10 (2)	reserved
11 (3)	reserved

fill_bits – These fill_bits accomplish byte-alignment of the ps_extension() data.

enable_ipdopd – The application of IPD and OPD parameters in the bit-stream is denoted by enable_ipdopd. If set (enable_ipdopd==%1) IPD and OPD parameters are sent, if disabled (enable_ipdopd==%0) no IPD and OPD parameters are sent for the current frame in the bit-stream. The quantization grid for both IPD and OPD is provided in Table 8.26. If no IPD or OPD data is sent in the bit-stream, all IPD and OPD parameters are set to 0 (i.e. index=0).

Table 8.26 - Quantization grid for IPD/OPD.

Index	0	1	2	3	4	5	6	7
Representation level	0	$\frac{\pi}{4}$	$\frac{\pi}{2}$	$\frac{3\pi}{4}$	π	$\frac{5}{4}\pi$	$\frac{3}{2}\pi$	$\frac{7}{4}\pi$

ipd_dt[e] – This flag describes for envelope index e, whether the IPD parameters are coded differentially over time (ipd_dt==%1) or over frequency (ipd_dt==%0). In the case iid_mode is different from the previous envelope (e-1), ipd_dt[e] shall have the value 0% forcing frequency differential coding.

ipd_data() – syntactic element containing IPD data.

opd_dt[e] – This flag describes for envelope index e, whether the OPD parameters are coded differentially over time (opd_dt==%1) or over frequency (opd_dt==%0). In the case iid_mode is different from the previous envelope (e-1), opd_dt[e] shall have the value 0% forcing frequency differential coding.

opd_data() – syntactic element containing OPD data.

reserved_ps – This bit is reserved and has a value %0.

iid_par_dt[e][b] – In case of differential coding of IID parameters over time (iid_dt[e]==%1), iid_par_dt[e][b] describes the IID index difference with respect to the b^{th} parameter position for envelope e-1. If no previous parameter is available, iid_par_dt[e][b] represents the IID index difference with respect to the decoded value 0 (i.e. index=0). The IID index iid_par[e][b] is determined as:

$$\text{iid\_par}[e][b] = \text{iid\_par}[e-1][b] + \text{iid\_par\_dt}[e][b]$$

where iid_par[e-1][b] represents the IID index of the previous envelope, e-1. The IID value *iid*[*b*] is obtained by using iid_par[e][b] as an index to Table 8.20 or Table 8.21, depending on iid_mode.

iid_par_df[e][b] – In case of differential coding of IID parameters over frequency (iid_dt[e]==%0), iid_par_df[e][b] describes the IID difference with respect to the (b-1)th parameter in envelope e. If no previous parameter is available, iid_par_df[e][b] represents the IID difference with respect to the decoded value 0 (i.e. index=0). The IID index iid_par[e][b] is determined as:

$$\text{iid\_par}[e][0] = \text{iid\_par\_df}[e][0]$$
$$\text{iid\_par}[e][b] = \text{iid\_par}[e][b-1] + \text{iid\_par\_df}[e][b] \qquad \text{for} \quad b > 0$$

where iid_par[e][b-1] represents the IID index of the previous IID value for envelope e. The IID value *iid*[*b*] is obtained by using iid_par[e][b] as an index to Table 8.20 or Table 8.21, depending on iid_mode.

icc_par_dt[e][b] – In case of differential coding of ICC parameters over time (icc_dt[e]==%1), icc_par_dt[e][b] describes the difference with respect to the b^{th} parameter position for envelope e-1. If no previous parameter is available, icc_par_dt[e][b] represents the ICC difference with respect to the decoded value 1 (i.e. index=0). The ICC index icc_par[e][b] is determined as:

$$\text{icc\_par}[e][b] = \text{icc\_par}[e-1][b] + \text{icc\_par\_dt}[e][b]$$

where icc_par[e-1][b] represents the ICC index of the previous envelope, e-1. The ICC value ρ[*b*] is obtained by using icc_par[e][b] as an index to Table 8.23.

icc_par_df[e][b] – In case of differential coding of ICC parameters over frequency (icc_dt[e]==%0), icc_par_df[e][b] describes the ICC difference with respect to the (b-1)th parameter for envelope e. If no previous parameter is available, icc_par_df[e][b] represents the ICC difference with respect to the decoded value 1 (i.e. index=0). The ICC index icc_par[e][b] is determined as:

$$\text{icc\_par}[e][0] = \text{icc\_par\_df}[e][0]$$
$$\text{icc\_par}[e][b] = \text{icc\_par}[e][b-1] + \text{icc\_par\_df}[e][b] \qquad \text{for} \quad b > 0$$

where icc_par[e][b-1] represents the ICC index of the previous ICC value for envelope e. The ICC value ρ[*b*] is obtained by using icc_par[e][b] as an index to Table 8.23.

ipd_par_dt[e][b] – In case of differential coding of IPD parameters over time (ipd_dt[e]==%1), ipd_par_dt[e][b] describes the IPD difference with respect to the b^{th} parameter position for envelope e. If no previous parameter is available, ipd_par_dt[e][b] represents the IPD difference with respect to the decoded value 0 (i.e. index=0). Note: for IPD parameters modulo-8 differential coding is applied. The IPD index ipd_par[e][b] is determined as:

$$\text{ipd\_par}[e][b] = \text{mod}(\text{ipd\_par}[e-1][b] + \text{ipd\_par\_dt}[e][b], 8)$$

where ipd_par[e-1][b] represents the IPD index of the previous envelope, e-1. The IPD value *ipd*[*b*] is obtained by using ipd_par[e][b] as an index to Table 8.26.

ipd_par_df[e][b] – In case of differential coding of IPD parameters over frequency (ipd_dt[e]==%0), ipd_par_df[e][b] describes the IPD difference with respect to the (b-1)th parameter for envelope e. If no previous parameter is available, ipd_par_df[e][b] represents the IPD difference with respect to the decoded

value 0 (i.e. index=0). Note: for IPD parameters modulo-8 differential coding is applied. The IPD index ipd_par[e][b] is determined as:

$$\text{ipd\_par}[e][0] = \text{ipd\_par\_df}[e][0]$$
$$\text{ipd\_par}[e][b] = \text{mod}(\text{ipd\_par}[e][b-1] + \text{ipd\_par\_df}[e][b], 8) \qquad \text{for} \quad b > 0$$

where ipd_par[e][b-1] represents the IPD index of the previous IPD value for envelope e. The IPD value *ipd*[*b*] is obtained by using ipd_par[e][b] as an index to Table 8.23.

opd_par_dt[e][b] – In case of differential coding of OPD parameters over time (opd_dt[e]==%1), opd_par_dt[e][b] describes the OPD difference with respect to the b^{th} parameter position for envelope (e-1). If no previous parameter is available, opd_par_dt[e][b] represents the OPD difference with respect to the decoded value 0 (i.e. index=0). Note: for OPD parameters modulo-8 differential coding is applied. The OPD index opd_par[e][b] is determined as:

$$\text{opd\_par}[e][b] = \text{mod}(\text{opd\_par}[e-1][b] + \text{opd\_par\_dt}[e][b], 8)$$

where opd_par[e-1][b] represents the OPD index of the previous envelope, e-1. The OPD value *opd*[*b*] is obtained by using opd_par[e][b] as an index to Table 8.26.

opd_par_df[e][b] – In case of differential coding of OPD parameters over time (opd_dt[e]==%0), opd_par_dt[e][b] describes the OPD difference with respect to the $(b\text{-}1)^{th}$ parameter position for envelope e. If no previous parameter is available, opd_par_df[e][b] represents the OPD difference with respect to the decoded value 0 (i.e. index=0). Note: for OPD parameters modulo-8 differential coding is applied. The OPD index opd_par[e][b] is determined as:

$$\text{opd\_par}[e][0] = \text{opd\_par\_df}[e][0]$$
$$\text{opd\_par}[e][b] = \text{mod}(\text{opd\_par}[e][b-1] + \text{opd\_par\_df}[e][b], 8) \qquad \text{for} \quad b > 0$$

where opd_par[e][b-1] represents the OPD index of the previous OPD value for envelope e. The OPD value *opd*[*b*] is obtained by using opd_par[e][b] as an index to Table 8.23.

8.5.3 Indexing of sub-frames

In the case differential coding is applied from one sub-frame to the next, a negative sub-frame index sf of frame *k* might be indexed. In case this occurs, the negative sub-frame shall be corrected according to

sf = sf + nrof_subframes.

Note that the sub-frame index thus obtained resides in frame *k-1*. Similarly, in the case that sf is larger than nrof_subframes, the sub-frame shall be corrected according to

sf = sf – nrof_subframes.

The sub-frame index thus obtained resides in frame *k+1*.

8.6 Decoding processes

The parametric stereo decoder is illustrated in Figure 8.1. After de-formatting the bit-stream, the monaural signal M is reconstructed as the combination of transients, sinusoids and noise. Subsequently the stereo parameters are used to reconstruct the left and right signal from the monaural encoded signal. For dual-mono and mono, the parametric stereo decoder is not used.

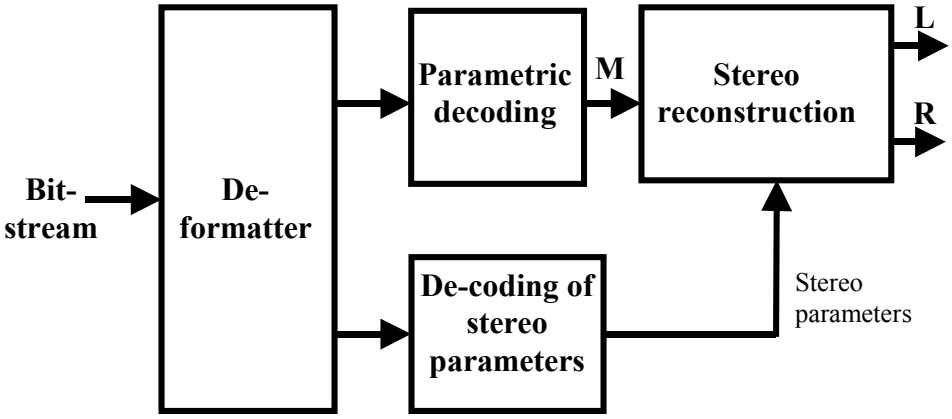

Figure 8.1 - Illustration of the integration of the stereo coding tools into the parametric decoder

The parametric decoder consists of three decoders: the transient decoder (8.6.1), the sinusoidal decoder (8.6.2) and the noise decoder (8.6.3). The decoded signal is obtained by summing the output of the three decoders per frame per channel. In the description of the decoders in the parametric decoder, the indexing of the sub-frame sf and channel ch is occasionally omitted for clarity.

8.6.1 Transients

Two types of transients are defined, a step transient and a transient of the Meixner type. The decoding of the step transient only involves the interpretation of the position. For the Meixner type, a parameterized envelope $g[n]$ and a number of sinusoids need to be decoded.

8.6.1.1 Step transient

A step transient does not generate a signal of its own (as the Meixner transient does), but it is used to modify the window shape for synthesizing sinusoidal and noise components (see sections 8.6.2.4 and 8.6.3).

8.6.1.2 Meixner transient

For the decoding of the Meixner transient, first the envelope needs to be generated. For the envelope the following parameters are required: the start position t_{loc}, the onset slope indicated by t_b_par (b parameter) and the decaying slope represented by t_chi_par (ξ parameter).

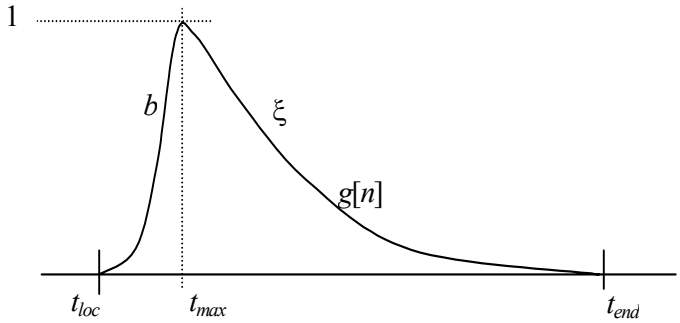

Figure 8.2 - The Meixner envelope is described by the function $g[n]$. The attack slope is controlled by the parameter b. The decay is controlled by the parameter ξ

The start time of the envelope, t_{loc}, is referred to as $n = 0$ for ease of explanation. This envelope $g[n]$ is generated according to

$$g[0] = \frac{\left(1-\xi^2\right)^{\frac{b}{2}}}{a_{\max}},$$

$$g[n] = g[n-1]\cdot\xi\cdot\sqrt{\frac{b+n-1}{n}},$$

for n=1 up to and including t_{end}. The end position of the transient window t_{end} is defined below. The maximum a_{max} at position t_{max} is given by the approximations

$$t_{\max} = \frac{b-1}{-2\cdot\log_e(\xi)},$$

$$a_{\max} = \sqrt{\frac{-2\cdot\log_e(\xi)}{\Gamma(b)}}\cdot\left(\frac{b-1}{e}\right)^{\frac{b-1}{2}}.$$

These complex expressions, especially that for a_{max}, have been evaluated for the allowed values of t_b_par and t_chi_par, and are tabulated below.

Table 8.27 - t_{max} for all possible values of t_b_par and t_chi_par.

t_{max}		t_b_par			
		0	1	2	3
t_chi_par	0	15	30	45	59
	1	20	39	57	75
	2	30	56	79	100
	3	57	96	126	150

Table 8.28 - a_{max} for all possible values of t_b_par and t_chi_par.

a_{max}		t_b_par			
		0	1	2	3
t_chi_par	0	0.152713500109658	0.131630525645664	0.120142673294398	0.112550174511598
	1	0.132843681407528	0.115639700421076	0.106510539071702	0.100663024431527
	2	0.109279971016712	0.0971964875412947	0.0909719057150294	0.0872632874594248
	3	0.0797175749717262	0.0744985442180281	0.0723059623257423	0.0715041477354716

The transient needs to be synthesized until t_{end}. This position t_{end} is defined as the end of the second complete subsequent sub-frame after the transient position t_{loc}. This is illustrated in Figure 8.3.

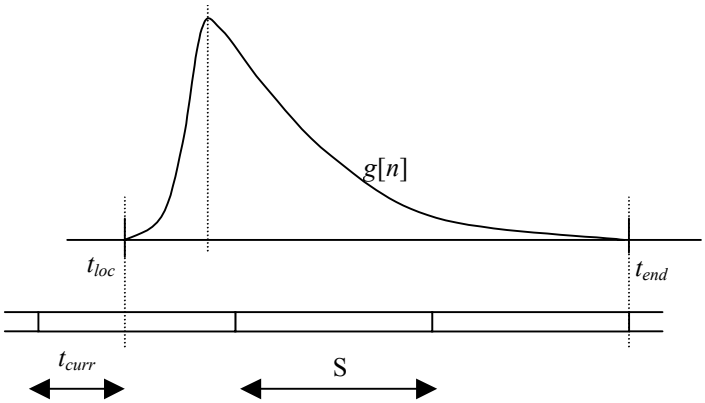

Figure 8.3 - The end position t_end is located at the end of the second complete sub-frame after the transient position t_loc

$$t_{end} = 3 \cdot \mathrm{S} - t_{loc} - 1$$

Note that the counting of samples in a sub-frame starts at 0. If the transient starts exactly at the beginning of a sub-frame (t_{loc} = 0), then t_{end} = 3*S-1.

8.6.1.3 Sinusoids under envelope

The resulting representation of the transient is obtained by combining the envelope and the sinusoids according to

$$t[n] = g[n] \cdot \sum_{i=1}^{t\_nrof\_sin} ta_q[i] \cdot \cos(tf_q[i] \cdot n + tp_q[i]),$$

for n=0 up to and including t_{end}.

8.6.2 Sinusoids

8.6.2.1 Linking

A refresh frame, indicated by refresh_sinusoids == %1, is used to indicate whether the frame is starting with absolute values for all continuations or whether the frame starts with differentially coded continuations. For the birth of each sinusoidal track in a frame, s_cont is supplied in the bit-stream to signal the number of sub-frames the track continues after the current sub-frame in this frame. In case the track continues in the first sub-frame of the next frame, 1 is added to that number. If the track continues beyond the first sub-frame of the next frame, 2 is added to that number. Based on this information, the decoder is implicitly able to link the parameters that belong to a track.

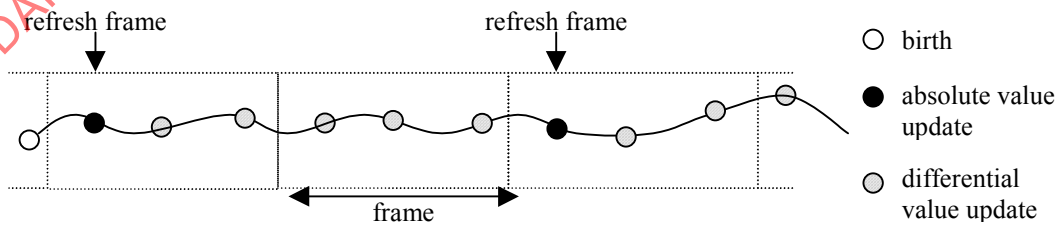

Figure 8.4 - Example for updating of a track. For every other frame, the track continuations are updated by their absolute values. In the case decoding starts at refresh_sinusoids==%0, the differential updates will be ignored until a refresh_sinusoids==%1 or a birth is received

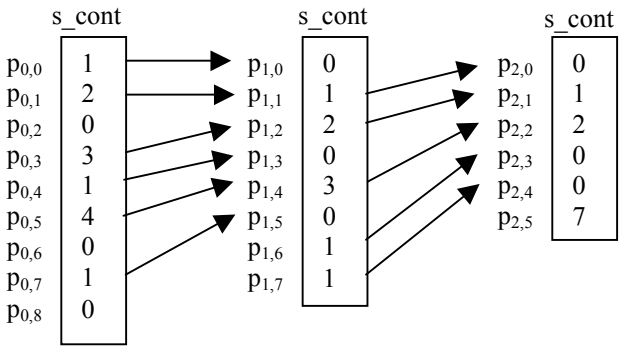

Figure 8.5 - Illustration of how the signaling of continuations operates. Parameter $p_{i,j}$ indicates the parameters (f, a and φ) of the j^{th} sinusoidal component in the i^{th} sub-frame. The information s_ cont that is sent along with these parameters provides the information on the evolution of the corresponding component in the next (sub)frame. In the above example, in sub-frame 1, components $p_{1,0}$, $p_{1,1}$, $p_{1,2}$, $p_{1,3}$, $p_{1,4}$ and $p_{1,5}$ are continued from sub-frame 0. Components $p_{1,0}$, $p_{1,3}$ and $p_{1,5}$ will die in sub-frame 1 and components $p_{1,6}$ and $p_{1,7}$ are new births

s_ cont is filled in the following order:

1) Continuations

2) Births (sorted on frequency in ascending order)

When going from one sub-frame to the next, the decoder keeps track of the number of continuations, s_nrof_continuations[sf]. The number of continuations present in sub-frame sf+1 can be directly derived from the number of entries in s_cont[sf] unequal to zero. For the first sub-frame of a frame, s_nrof_continuations is read from the bit-stream, to enable random access.

The total amount of sinusoidal components in sub-frame sf, s_nrof_sin[sf], is calculated as

sf == 0:

$$s\_nrof\_sin[0] = s\_nrof\_continuations[0][ch] + s\_nrof\_births[0][ch],$$

sf > 0:

$$s\_nrof\_sin[sf] = \sum_{i=0}^{max\_nrof\_sinusoids-1} (s\_cont[sf-1][ch][i] > 0) + s\_nrof\_births[sf][ch].$$

8.6.2.2 Decoding of sinusoidal parameters

In the description below, we assume to have a sinusoidal track of length κ, in sub-frames *sf* = [*K, K*+κ-*1*]. For births of the track (*sf* = *K*), the frequency and phase for sinusoid index *n* are represented by $f_q[K][ch][n]$ and $sp_q[K][ch][n]$ respectively. For continuations, in order to derive the frequency and phase information for a sub-frame, the representation levels together with tracking information is required. Figure 8.6 shows the decoder structure for continuations.

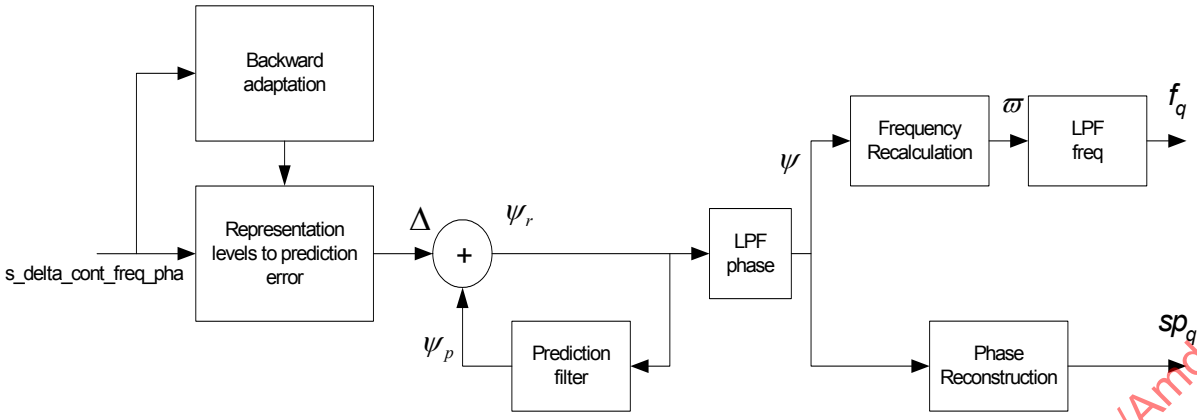

Figure 8.6 - The phase/frequency decoding system for continuations

For a continuation (*sf* = [*K+1, K+κ-1*]), the representation levels s_delta_cont_freq_pha[sf][ch][n] are converted to a quantized prediction error Δ[*sf*][*ch*[*n*] using Table 8.30 with index == 2. The quantized prediction error Δ[*sf*][*ch*][*n*] is subsequently multiplied by scale factor *c*[*sf*][*ch*][*n*]. For the first continuation, *c*[*K+1*][*ch*][*p*], where *p* represents the sinusoid index in sub-frame *K*+1, depends on the frequency of the birth, i.e. $f_q[K]$. Table 8.29 shows the value of the scale factor *c* for the possible frequency ranges of $f_q[K]$.

Table 8.29 - Scale factor table

Freq. range (in Hertz)	Scale factor *c*[*K*+1]
[0 – 500] Hz	1/8
<500-1000] Hz	1/4
<1000-4000] Hz	1/2
<4000-22050] Hz	1

For other continuations, *sf* = [*K+2, K+κ-1*], *c* is modified according to the received representation levels along the track by means of the block “Backward Adaptation”. If s_delta_cont_freq_pha[sf][ch][n] is either 1 or 2 (inner level) for sub-frame *sf*, then *c* for sub-frame *sf+1* is set to

$$c[sf+1][ch][p] = c[sf][ch][n] \cdot 2^{-1/4}.$$

If s_delta_cont_freq_pha[sf][ch][n] equals 0 or 3 (outer level), then *c* for sub-frame *sf+1* is set to

$$c[sf+1][ch][p] = c[sf][ch][n] \cdot 2^{1/2}.$$

In order to avoid very small or very large entries in prediction error, the adaptation is only done if the absolute value of the inner level, 0.75*c*[*sf+1*][*ch*][*p*], is between π/128 and 3π/8. In the case were the inner level is less than or equal to π/128 or greater than or equal to 3π/8, the scale factor *c*[*sf+1*][*ch*][*p*] is set to *c*[*sf*][*ch*][*n*]. This is illustrated in Table 8.30.

After having obtained the quantised prediction error Δ, the output of the prediction filter is added to it, resulting in the unwrapped phase ψ_r.

$$\psi_r[sf][ch][n] = \psi_p[sf][ch][n] + \Delta[sf][ch][n] \cdot c[sf][ch][n].$$

A second-order predictor is used. The input-output behaviour of the filter is

$$\psi_p[sf+1][ch][p] = 2 \cdot \psi_r[sf][ch][n] - \psi_r[sf-1][ch][q],$$

where *q* is the sinusoid index in sub-frame *sf*-1, ψ_r is the input and ψ_p is the output of the prediction filter.

To initialize the prediction filter we need one value of history for the input $\psi_r[K-1]$, where *K* is the sub-frame index of the birth of the track. Since this value is not available, we assume that the frequency is constant at sub frame *(K-1)*. For birth of a track we have the frequency and phase information available, so we can calculate the input at *(K-1) and K* according to,

$$\psi_r[K-1][ch][n] = sp_q[K][ch][n] - f_q[K][ch][n] \cdot S,$$
$$\psi_r[K][ch][n] = sp_q[K][ch][n].$$

where *S* represents the update interval. The unwrapped phases are low-pass filtered by block LPF-phase. This is done as follows:

$$\psi[sf][ch][n] = 0.25 \cdot \psi_r[sf+1][ch][p] + 0.5 \cdot \psi_r[sf][ch][n] + 0.25 \cdot \psi_r[sf-1][ch][q],$$

where *sf* is the sub-frame index along a track. At the end of a track (*sf* = *K*+κ-*1*), the following rule is applied:

$$\psi[sf][ch][n] = \psi_r[sf][ch][n].$$

The reconstructed phases are derived from the smoothed unwrapped phases as follows:

$$sp_q[sf][ch][n] = \mathrm{mod}(\psi[sf][ch][n] + \pi, 2\pi) - \pi \ .$$

To obtain the frequency, the unwrapped phases have to be differentiated along the track. The differentiation is implemented by an approximation. The frequency is obtained by:

$$\varpi[sf][ch][n] = \left(\psi[sf][ch][n] - \psi[sf-1][ch][q]\right)\frac{2}{S} - \varpi[sf-1][ch][q],$$

where S is the update interval and $\varpi[K][ch][n] = f_q[K][ch][n]$. As the birth phase and birth frequency are known in the decoder, the frequencies ϖ of the subsequent frames are calculated. In order to attenuate the noisy signal that is introduced by this differentiation, a low-pass filter is applied on the frequencies (LPF-freq):

$$f_q[sf][ch][n] = 0.25 \cdot \varpi[sf+1][ch][p] + 0.5 \cdot \varpi[sf][ch][n] + 0.25 \cdot \varpi[sf-1][ch][q].$$

For the first continuation of a track (*sf*=*K*+*1*), the definition is changed to:

$$f_q[K+1][ch][p] = 0.5 \cdot \varpi[K+1][ch][p] + 0.5 \cdot \varpi[K][ch][n].$$

Also the last frequency in the track (*sf* = *K*+κ-*1*) is obtained in a different manner:

$$f_q[sf][ch][n] = 0.5 \cdot \varpi[sf][ch][n] + 0.5 \cdot \varpi[sf-1][ch][q].$$

For tracks of length κ=2 the continuation is calculated according to

$$f_q[sf][ch][n] = 0.5 \cdot \varpi[sf-1][ch][q] + 0.5 \cdot \varpi[sf][ch][n].$$

In this way, the phases and frequencies are obtained from the representation levels s_delta_cont_freq_pha.

In refreshes frames, the following procedure applies. If *sf*=[*K, …, K+R, …, K+κ-1*]. The sub-frame, *K+R* is the first sub-frame of a frame with refresh_sinusoids == %1. Sub-frame (*K+R*-1) is the last sub-frame of a frame with refresh_sinusoids_next_frame == %1. The phase and frequency values for sub-frame *K* up to and including sub-frame *K+R*-1 are obtained as described above, as if the track ends at sub-frame (*K+R*-1). The

phase and frequency values for sub-frame *K+R* up to and including sub-frame (*K*+κ-1) are obtained as described above, as if sub-frame *K+R* is a birth. For the initialization of the quantized prediction error Δ, s_adpcm_grid is used as index to Table 8.30 and *c*[*K+R*][*ch*][*n*] = 1.

Table 8.30 – Quantized prediction error Δ

Δ	s_delta_cont_freq_pha			
index	0	1	2	3
0	-4.2426	-1.0607	1.0607	4.2426
1	-3.5676	-0.8919	0.8919	3.5676
2	-3.0000	-0.7500	0.7500	3.0000
3	-2.5227	-0.6307	0.6307	2.5227
4	-2.1213	-0.5303	0.5303	2.1213
5	-1.7838	-0.4460	0.4460	1.7838
6	-1.5000	-0.3750	0.3750	1.5000
7	-1.2613	-0.3153	0.3153	1.2613
8	-1.0607	-0.2652	0.2652	1.0607
9	-0.8919	-0.2230	0.2230	0.8919
10	-0.7500	-0.1875	0.1875	0.7500
11	-0.6307	-0.1577	0.1577	0.6307
12	-0.5303	-0.1326	0.1326	0.5303
13	-0.4460	-0.1115	0.1115	0.4460
14	-0.3750	-0.0938	0.0938	0.3750
15	-0.3153	-0.0788	0.0788	0.3153
16	-0.2652	-0.0663	0.0663	0.2652
17	-0.2230	-0.0557	0.0557	0.2230
18	-0.1875	-0.0469	0.0469	0.1875
19	-0.1577	-0.0394	0.0394	0.1577
20	-0.1326	-0.0331	0.0331	0.1326
21	-0.1115	-0.0279	0.0279	0.1115

Note that in the case phase_jitter_present is set to %1 and phase and frequency are obtained through ADPCM decoding, no phase jitter is applied.

8.6.2.3 Synthesis of sinusoids for segments without a transient

The sinusoidal parameters are used to synthesize the sinusoidal components. This is done on a segment basis, a segment consisting of L samples.

Synthesis uses a 50% overlap and add strategy. In order to synthesize a sub-frame, the parameters of the previous sub-frame need to be available at the start of a new frame. This means that the parameters of the last sub-frame in the previous frame need to be available.

The actual synthesis of a sinusoid is according to

$$s_i[n] = sa_q[i] \cdot \cos\left(f_q[i] \cdot \left(n - \frac{L-1}{2} \right) + sp_q[i] \right), \quad n = [0, L-1]$$

Note that the phase information sp_q is defined for the middle of the segment (=(L-1)/2). For original phase, sp_q is calculated from s_phi (see section 8.5.2). For continuations, the phase is calculated as described in section 8.6.2.2.

In overlap and add the following, amplitude complementary symmetric window is chosen

$$w_s[n] = 1/2 - 1/2 \cos\left(\frac{\pi(2n+1)}{L}\right), \quad n = [0, L-1].$$

A segment of length L is obtained by

$$\tilde{s}_{sf}[n] = w_s[n] \sum_{i=0}^{s\_nrof\_sin[sf]} s_i[n], \quad n = [0, L-1]$$

The sinusoidal contribution for sub-frame sf is then computed using the contribution from the previous sub-frame according to

$$s_{sf}[n] = \tilde{s}_{sf-1}[S+n] + \tilde{s}_{sf}[n], \quad n = [0, S-1]$$

8.6.2.4 Synthesis of sinusoids for segments without a transient

For sinusoids with a frequency lower than 400Hz only the window $w_s[n]$ defined in the previous section should be used, even at a segment containing a transient.

For other sinusoids, different window shapes apply for synthesis of a segment, depending on the course of a component (continuation, birth, or death). As a result of transient positions, some of these windows are modified. All the possibilities are illustrated in Figure 8.7.

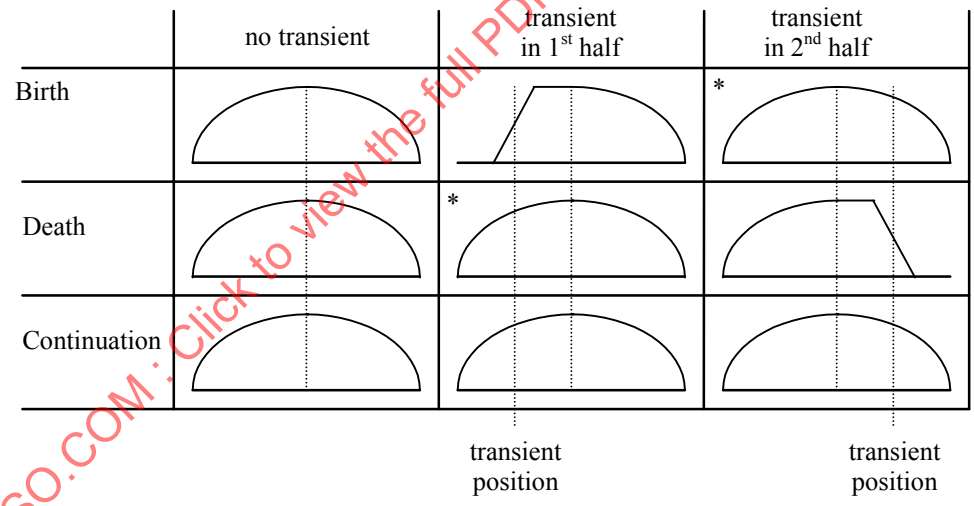

Figure 8.7 - Possible synthesis windows $w_s[n]$ for births, deaths and continuations, dependent on transient occurrence and position. For the situations denoted by * the transient and sinusoidal component have no relation.

This figure shows that there are two situations where the default window shape may not be used:

- a birth in combination with a transient located in the first half of the segment
- a death in combination with a transient in the second half of the segment.

8.6.2.4.1 Birth and transient in first half of segment

For births, a fade in window will be used to prevent substantial energy leaking into the interval prior to the transient position. In Figure 8.7 only a coarse shape of the window is shown. Figure 8.8 shows a more detailed view of the window.

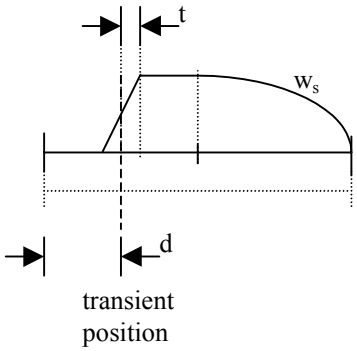

Figure 8.8 - Synthesis window shape for a birth where a transient occurs in the first half of the segment

The following expression exactly describes the window $w_s[n]$

$$w_s[n] = \begin{cases} 0 & ,0 \le n < d-10 \\ \dfrac{n-d+11}{22} & ,d-10 \le n \le d+10 \\ 1 & ,d+10 < n < S \\ \frac{1}{2} - \frac{1}{2}\cos\left(\pi \cdot \dfrac{2n+1}{L}\right) & ,S \le n < L \end{cases}$$

In the special case that the transient position is located within 10 samples from the “edge”, the slope is moved (see Figure 8.9).

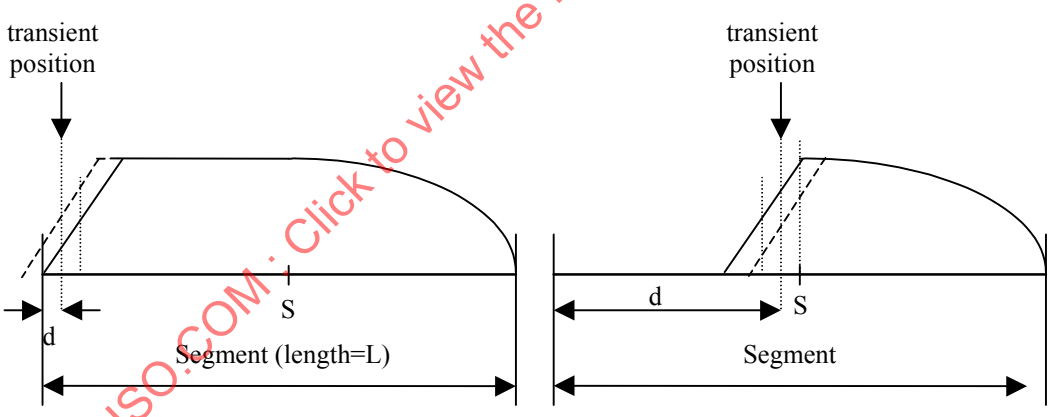

Figure 8.9 - Illustration of the fade-in window in the cases where the transient is located less than 10 samples from the “edge”; the slope is shifted to start at the “edge” (left picture) or end at the “edge” (right picture)

For the left picture in Figure 8.9 this means that if $0 \le d < 10$, then use $d = 10$ for determining the window function $w_s[n]$. For the right picture in Figure 8.9 this means that if $S\text{-}10 \le d < S$, then use $d = S–11$ for determining the window function $w_s[n]$.

8.6.2.4.2 Death and transient in second half of segment

In Figure 8.7 only a coarse shape of the window is shown. Figure 8.10 shows a more detailed view of the window.

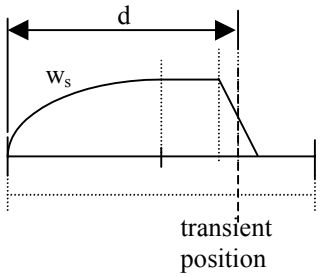

Figure 8.10 - Synthesis window shape for a death where a transient occurs in the second half of the segment

The following expression describes the window $w_s[n]$

$$w_s[n] = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos\left(\pi \cdot \frac{2n+1}{L}\right) & ,0 \le n < S \\ 1 & ,S \le n < d-10 \\ \frac{d-n+11}{22} & ,d-10 \le n \le d+10 \\ 0 & ,d+10 < n < L \end{cases}$$

In the special case that the transient position is located within 10 samples from the “edge”, the slope is moved (see Figure 8.11).

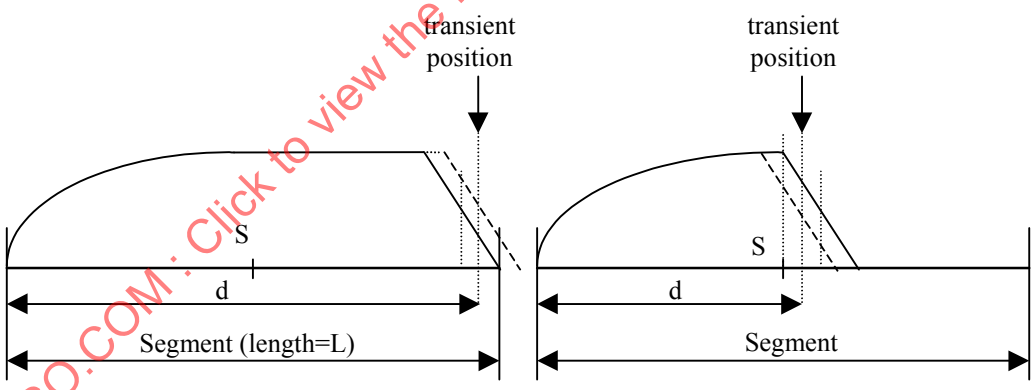

Figure 8.11 - Illustration of the fade-out window in the cases where the transient is located less than 10 samples from the “edge”; the slope is shifted to end at the “edge” (left picture) or start at the “edge” (right picture)

For the left picture in Figure 8.11 this means that if L-10 ≤ d < L , then use d = L-11 for determining the window function $w_s[n]$. For the right picture in Figure 8.11 this means that if S ≤ d < S + 10, then use d = S+10 for determining the window function $w_s[n]$.

8.6.3 Noise

The noise is synthesized in intervals of 4 sub-frames or 2L samples. The model for noise synthesis consists of a pseudo random number generator, a temporal envelope adjuster, a window mechanism for overlap and add and an IIR filter.

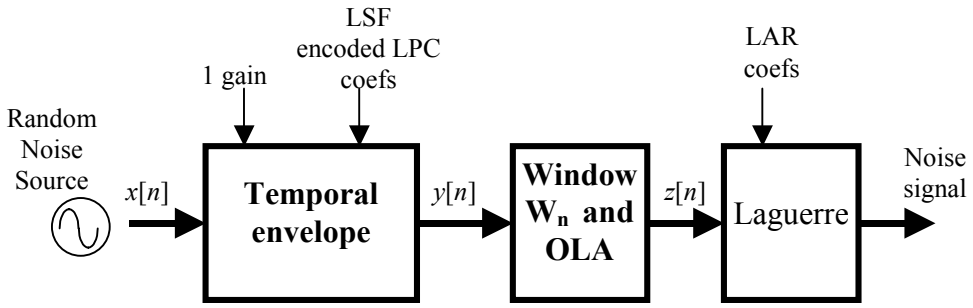

Figure 8.12 - Noise decoder. Time intervals of random noise excitations are adjusted by temporal envelopes which are windowed, overlap/added and subsequently filtered by the Laguerre filter, resulting in the noise signal. Per iteration one interval of noise is created for *n* = [0, 2L-1]

The temporal envelope *H*[*n*] is represented by using a single gain and a number of line spectral frequencies (LSFs) representing the LPC coefficients. Both the gain and the LSFs are updated once every 4 sub-frames (2L samples). The Laguerre coefficients are represented by LAR parameters. These are updated once per 2 sub-frames (L samples), which is twice as low update-rate as used for the temporal envelope, see Figure 8.13.

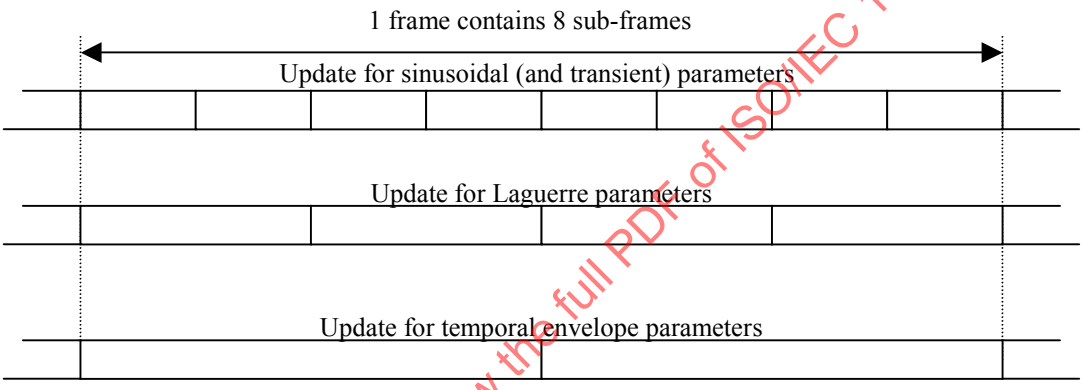

Figure 8.13 - The update rate for the Laguerre parameters is for every two sub-frames (2S = L). The temporal parameters are updated every four sub-frames (4S = 2L)

In order to prevent discontinuities, intervals that are modified by a temporal envelope have an overlap of 25%. In the overlap region a Hanning window is employed, see Figure 8.14. Note that the first 4 generated sub-frames of 2L samples start with a fade-in using the Hanning window.

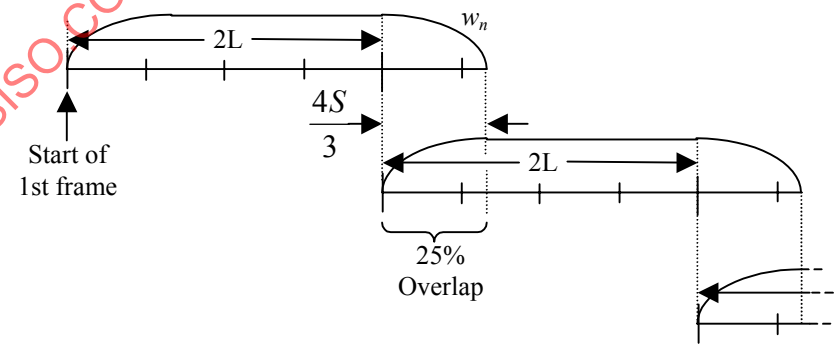

Figure 8.14 - Overlap and add. In the overlap a half Hanning window h is used. In the intermediate part the window is set to 1. The first 4 sub-frames are starting with a fade-in realised by the Hanning window

The window w_n is defined as given below.

$$w_n[n] = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos\left(\frac{3\pi(2n+1)}{8S}\right) & , \quad n = [0, \frac{4S}{3} - 1], \\ 1 & , \quad n = [\frac{4S}{3}, 4S - 1], \\ w_n[\frac{16S}{3} - 1 - n] & , \quad n = [4S, \frac{16S}{3} - 1]. \end{cases}$$

The operations performed by each of the individual processes is discussed in more detail in the following sections.

8.6.3.1 Noise generation

The noise is generated by applying a pseudo random number generator, defined by a linear congruential sequence U

$U[n+1] = \text{mod}\ (a*U[n]+c, m)$,

where $U[0]$ is the starting value, a the multiplier, c the increment and m the modulus (with $m=2^{32}$). At the start of decoding the starting value is set to channel_number (0 = left, 1 = right), resulting in independent noise sources for each channel. For the generation of each following interval of noise, the starting value is set to the end value of the previous interval. The algorithm is listed below.

```
#define RAND_SCALE      (1/4294967296.0)
#define RAND_FACTOR     1664525L
#define RAND_OFFSET     1013904223L
double  noiseUDN(unsigned long *lp_seed)
{
    *lp_seed = (*lp_seed * RAND_FACTOR+RAND_OFFSET) & 0xFFFFFFFF;
    return *lp_seed * RAND_SCALE;
}
```

This algorithm returns a value $U = [0,1\rangle$. A normal distribution X is obtained by addition of 12 consecutive samples of the distribution (U-0.5). For the next sample of X, 12 new consecutive samples are used. Using this normal distribution X, by means of noise filtering, the spectral noise is generated.

In order to avoid discontinuities in the noise generation, overlapping time intervals are taken from the free running noise generator. This is realized by copying the seed values from the previous interval to the current interval during overlap. This is illustrated in Figure 8.15.

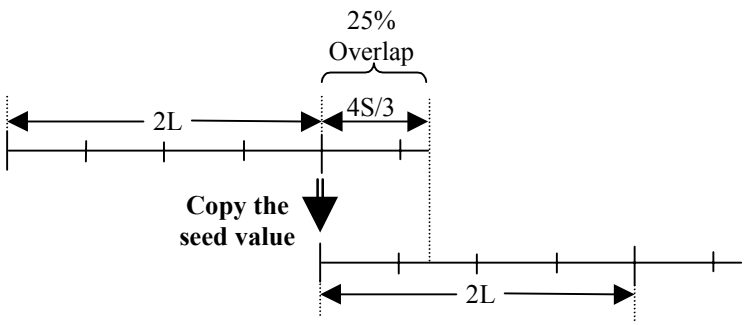

Figure 8.15 - The start-seed value used in the next interval is copied from the previous interval at the corresponding time location

8.6.3.2 Temporal envelope

The temporal envelope is applied to an interval of 2L+4S/3 samples, which are generated by the random noise generator. The temporal envelope shape is represented by the time domain equivalent of Line Spectral Frequencies, which are again a representation of LPC coefficients. In total n_nrof_lsf LSFs are encoded. An additional gain parameter is used to scale the entire envelope. Since the LSF intervals have an overlap of 25% there is a potential redundancy in LSF parameters in this overlap region. In the case this redundancy is present, only one set of LSFs, valid for two envelopes in the overlap region, is encoded. This situation is signalled by the parameter n_overlap_lsf. In the case of refresh_noise == %1, the first LSF and gain for that particular interval are encoded in absolute values by means of the parameters n_lsf and n_gain. Subsequent LSFs are encoded differentially w.r.t. each other, see Figure 8.16.

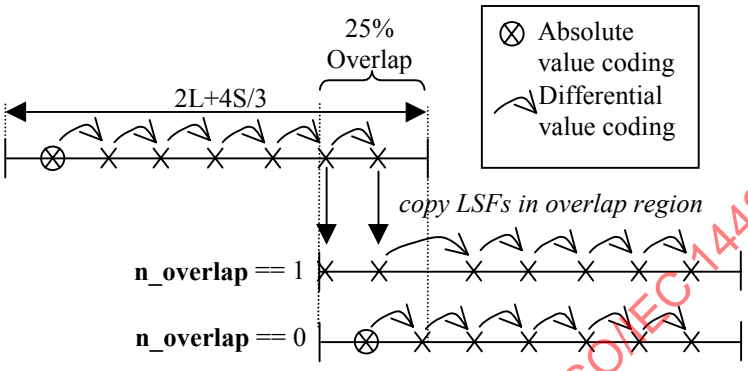

Figure 8.16 - Example for 7 LSF parameters. For the first interval, the first LSF parameter is encoded in its absolute value n_lsf. Subsequent LSF values in that frame are differentially encoded w.r.t. each other by means of n_delta_lsf. In the case n_overlap equals 1, all LSFs in the overlap region are copied to the next interval. Subsequent LSFs are again differentially encoded w.r.t. each other. In the case n_overlap equals 0, the LSFs are obtained similar to the first interval

In the case refresh_noise == %0, the gain parameter for that interval is encoded differentially w.r.t. the gain of the previous frame by means of the parameter n_delta_gain. The encoding of the LSF parameters in that situation depends on the setting of n_overlap_lsf. In the case refresh_noise == %0 and n_overlap_lsf == %0, the LSF are encoded as in the situation where refresh_noise == %1. In the case refresh_noise == %0 and n_overlap_lsf == %1, the number of LSF coefficients that overlap, n_nrof_overlap, is computed from the previous definition in channel ch according to

```
n_nrof_overlap = 0;
   for ( i = 0; i < n_nrof_lsf; i++) {
   if ( n_lsf[sf-4][ch][i] >= 192 ) n_nrof_overlap++
}
```

The LSF coefficients that overlap are copied from the previous definition according to

```
for ( i = 0, j = n_nrof_lsf – n_nrof_overlap; i < n_nrof_overlap; i++, j++)
{
   n_lsf[sf][ch][i] = n_lsf[sf-4][ch][j] - 192
}
```

8.6.3.2.1 Decoding the Gain and LSF parameters

The gain scales the entire temporal envelope. Since there are two envelopes per frame, the gain for the first and second temporal envelope are encoded in sf=0 and sf=4 respectively. The gain factor *G* which is actually applied to a temporal envelope is calculated as

$$G = \begin{cases} 0 & ngain_{rl} == 0 \\ 10^{\frac{ngain_{rl}-21}{20}} & \text{otherwise} \end{cases}.$$

The decoded LSFs, $nlsf_q$ are transformed to a-parameters using the following equations. First, all LSFs are transformed to positions on the unit circle

$$z[i] = e^{j \cdot nlsf_q[i]} \qquad i = [1, n\_nr\_of\_lsf].$$

These positions are then split into two polynomials:

$$z_P[i] = z[2i] \qquad i = \left[1, \left\lfloor \frac{n\_nr\_of\_lsf}{2} \right\rfloor\right],$$

$$z_Q[i] = z[2i-1] \qquad i = \left[1, \left\lceil \frac{n\_nr\_of\_lsf}{2} \right\rceil\right].$$

For both polynomials the complex conjugates are concatenated

$$z_P[i] = z_P^*\left[i - \left\lfloor \frac{n\_nr\_of\_lsf}{2} \right\rfloor\right] \qquad i = \left[1 + \left\lfloor \frac{n\_nr\_of\_lsf}{2} \right\rfloor, 2\left\lfloor \frac{n\_nr\_of\_lsf}{2} \right\rfloor\right],$$

$$z_Q[i] = z_Q^*\left[i - \left\lceil \frac{n\_nr\_of\_lsf}{2} \right\rceil\right] \qquad i = \left[1 + \left\lceil \frac{n\_nr\_of\_lsf}{2} \right\rceil, 2\left\lceil \frac{n\_nr\_of\_lsf}{2} \right\rceil\right].$$

The polynomials are calculated as following

$$p_P = \prod_{i=1}^{2\left\lfloor \frac{n\_nr\_of\_lsf}{2} \right\rfloor} (z - z_P[i]),$$

$$p_Q = \prod_{i=1}^{2\left\lceil \frac{n\_nr\_of\_lsf}{2} \right\rceil} (z - z_Q[i]).$$

In case n_nr_of_lsf is odd the polynomials are finally changed to

$$p_Q = p_Q(z+1)(z-1).$$

In case n_nr_of_lsf is even

$$p_P = p_P(z-1)$$
$$p_Q = p_Q(z+1).$$

The polynomial A(z) is given as

$$A(z) = \frac{p_P(z) + p_Q(z)}{2}.$$

Finally the envelope *H*[*n*], where *n* is the sample index, is calculated as

$$H[n] = \frac{G}{\left| A\left(e^{j\frac{n\pi}{2L+\frac{4S}{3}}} \right) \right|}, \quad n = \left[0{,}2L + \frac{4S}{3} - 1 \right].$$

The noise sequence is multiplied by the temporal envelope. Note, that in the case n_nr_of_lsf == 0, $H[n]$ is defined as

$$H[n] = G, \quad n = \left[0{,}2L + \frac{4S}{3} - 1 \right].$$

8.6.3.3 Noise filtering

The structure of the Laguerre synthesis filter, which is applied after the overlap-add operation, is given in Figure 8.17.

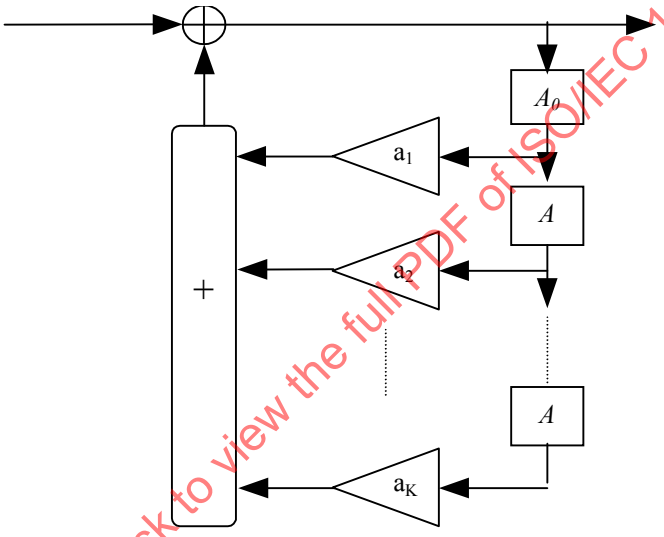

Figure 8.17 - Structure of Laguerre synthesis filter

The filter A_0 is a first-order filter, the filters A are first-order all-pass sections and λ is the so called Laguerre pole whose absolute value is less than 1.

$$A_0(z) = \sqrt{1-\lambda^2}\,\frac{z^{-1}}{1-z^{-1}\lambda}$$

$$A(z) = \frac{-\lambda + z^{-1}}{1-z^{-1}\lambda}$$

Note that a value of 0 for λ is equivalent to conventional LPC.

The parameters for the Laguerre filtering are updated every 2 sub-frames (L samples). In order to make sure that in the generation of the first sample already the desired spectral density is obtained, the initial filter states need to be set. This is realized by copying the final states after the generation of an interval into the initial states for the generation of the next interval. In case that refresh_noise == %0, or start of decoding, the initial filter states will be set to 0. So, using a first set of parameters, two consecutive sub-frames are filtered. For the next 2 sub-frames (with new parameters) the final filter-states resulting from the previous filter operation are used as initial states. For the updates of the LAR parameters differential coding is employed with respect to the LAR parameters of the previous interval.

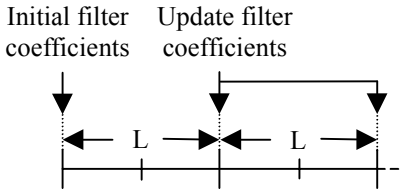

Figure 8.18 - In order to set the initial states of the filter, the final states after the generation of an interval of 2L samples are copied into the initial states for the generation of the next interval of 2L samples

In the bit-stream, for each sub-frame, coefficients for a Laguerre filter structure are encoded in a Log Area Ratio (LAR) notation. The first step of the reconstruction of the Laguerre parameters is the dequantisation of the LARs (see section 8.6.3.3.1). The parcor coefficients ('rfc') are obtained from the dequantised LARs according to the procedure depicted in section 8.6.3.3.2. The parcors are transformed to FIR coefficients (see section 8.6.3.3.3). The last step is the conversion of FIR coefficients back to the Laguerre coefficients *a* (see section 8.6.3.3.4).

8.6.3.3.1 Quantized LARs

The LAR coefficients in the denominator are de-quantized by multiplying the value encoded in the bit-stream with a constant Δ_{LAR} which is defined as

$$\Delta_{LAR} = \frac{dynr}{levels - 1},$$

where dynr=2*8 is the dynamic range of the LAR coefficients (from –8 to +8), and levels=2^{bits}-2, with bits=9, represents the number of representation levels.

8.6.3.3.2 Convert LARs into parcors

The following algorithm describes the conversion of m LAR coefficients $nlar_q$ into *m* (*m*=n_nrof_den) parcors 'rfc'.

```
for (i=0; i<m; i++)
{
    rfc[i]=(exp(nlar_q[i])-1)/(exp(nlar_q[i])+1)
}
```

8.6.3.3.3 Convert parcors into FIR coefficients

The following algorithm describes the conversion of m parcor coefficients 'rfc' into m+1 a-parameters 'p'.

```
for (k=0; k<m; k++)
{
   d[k] = –rfc[k];
   for (i=0;i<k;i++)
   {
        d[i] = tmp[i]+rfc[k]*tmp[k-i-1];
   }
   for (i=0;i<=k;i++)
   {
        tmp[i] = d[i];
   }
}
p[0] = 1.0;
for (k = 0; k < m; k++)
{
   p[k+1] = -d[k];
}
```

8.6.3.3.4 Convert FIR coefficients into Laguerre coefficients

The a-parameters *p* are converted back to Laguerre coefficients *a* using the following algorithm

$$a'_M = p_M ,$$
$$a'_m = p_m - a'_{m+1}\lambda ,$$

where $k = [\text{n\_nrof\_den} - 1 .. 0]$ and

$$a_m = -\frac{a'_m}{a'_0}\sqrt{1-\lambda^2} .$$

8.6.4 Parametric stereo

8.6.4.1 Stereo parameters

Three different types of stereo parameters are used in representing the stereo image. For in total a maximum of 34 bands, one set of stereo parameters is available per band.

1) The inter-channel intensity difference, or IID, defined by the relative levels of the band-limited signal.

2) The inter-channel and overall phase differences, IPD and OPD, defining the phase behaviour of the band-limited signal.

3) The inter-channel coherence ICC, defining the (dis)similarity of the left and right band-limited signal.

Figure 8.19 diagrams the processing chain of the parametric stereo decoder.

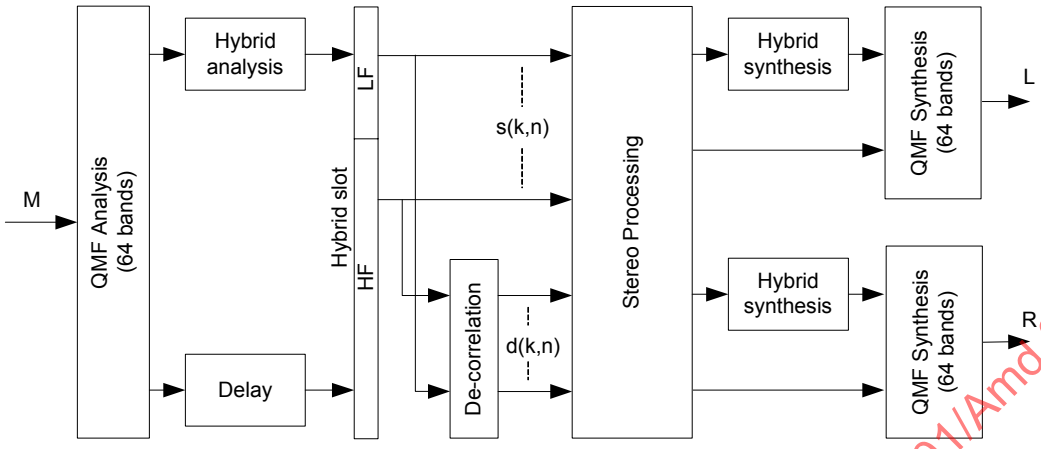

Figure 8.19 - QMF based parametric stereo synthesis

The input to the parametric stereo decoder consists of the monaural parametric generated signal M as obtained through transient, sinusoidal and noise synthesis. The output consists of the left and right stereo representation respectively. In the next paragraphs each block will be detailed.

8.6.4.2 QMF analysis filterbank

This filterbank is identical to the 64 complex QMF analysis filterbank as defined in ISO/IEC 14496-3/AMD1:2003, subclause 4.B.18.2. However, in the equation for matrix M(k,n) and in Figure 4.B.20, the term "(2*n+1)" has to be substituted by "(2*n-1)". The input to the filterbank are blocks of 64 samples of the monaural synthesized signal M. For each block the filterbank outputs one slot of 64 QMF samples.

8.6.4.3 Low frequency filtering

The lower QMF subbands are further split in order to obtain a higher frequency resolution enabling a proper stereo analysis and synthesis for the lower frequencies. Depending on the number of stereo bands, two hybrid configurations have been defined. See Table 8.31 for an overview of the splits and the type of filter that is used to make the split.

Table 8.31 - Overview of low frequency split for the available configurations.

Configuration, number of stereo bands	QMF subband p	Number of bands Q^p	Filter
10, 20	0	8	Type A
	1	2	Type B
	2	2	
34	0	12	Type A
	1	8	
	2	4	
	3	4	
	4	4	

$$TypeA: G_q^p = g^p[n]\exp(j\frac{2\pi}{Q^p}(q+\frac{1}{2})(n-6)),$$

$$TypeB: G_q^p = g^p[n]\cos(\frac{2\pi}{Q^p}q(n-6)),$$

where g^p represents the prototype filters in QMF subband p. Q^p represents the number of sub-subbands in QMF subband p, q the sub-subband index in QMF channel p and n the time index. The prototype filters are all of length 13 and have a delay of 6 QMF samples. The prototype filters are listed in Table 8.32 and Table 8.33 for the 10,20 and the 34 stereo bands configuration respectively.

Table 8.32 - Prototype filter coefficients for the filters that split the lower QMF subbands for the 10 and 20 stereo bands configuration.

n	$g^0[n]$, Q^0 =8	$g^{1,2}[n]$, $Q^{1,2}$ =2
0	0.00746082949812	0
1	0.02270420949825	0.01899487526049
2	0.04546865930473	0
3	0.07266113929591	-0.07293139167538
4	0.09885108575264	0
5	0.11793710567217	0.30596630545168
6	0.125	0.5
7	0.11793710567217	0.30596630545168
8	0.09885108575264	0
9	0.07266113929591	-0.07293139167538
10	0.04546865930473	0
11	0.02270420949825	0.01899487526049
12	0.00746082949812	0

Table 8.33 - Prototype filter coefficients for the filters that split the lower QMF subbands for the 34 stereo bands configuration.

n	$g^0[n]$, Q^0 =12	$g^1[n]$, Q^1 =8	$g^{2,3,4}$, $Q^{2,3,4}$ =4
0	0.04081179924692	0.01565675600122	-0.05908211155639
1	0.03812810994926	0.03752716391991	-0.04871498374946
2	0.05144908135699	0.05417891378782	0
3	0.06399831151592	0.08417044116767	0.07778723915851
4	0.07428313801106	0.10307344158036	0.16486303567403
5	0.08100347892914	0.12222452249753	0.23279856662996
6	0.08333333333333	0.12500000000000	0.25000000000000
7	0.08100347892914	0.12222452249753	0.23279856662996
8	0.07428313801106	0.10307344158036	0.16486303567403
9	0.06399831151592	0.08417044116767	0.07778723915851
10	0.05144908135699	0.05417891378782	0
11	0.03812810994926	0.03752716391991	-0.04871498374946
12	0.04081179924692	0.01565675600122	-0.05908211155639

Figure 8.20 and Figure 8.21 illustrate the hybrid analysis and synthesis filterbank for the 10 and 20 stereo bands configuration respectively. Figure 8.22 and Figure 8.23 illustrate the hybrid analysis and synthesis filterbank for the 34 stereo bands configuration respectively. Note that for the 10 and 20 stereo bands configuration, sub-subbands have been combined into a single sub-subband.

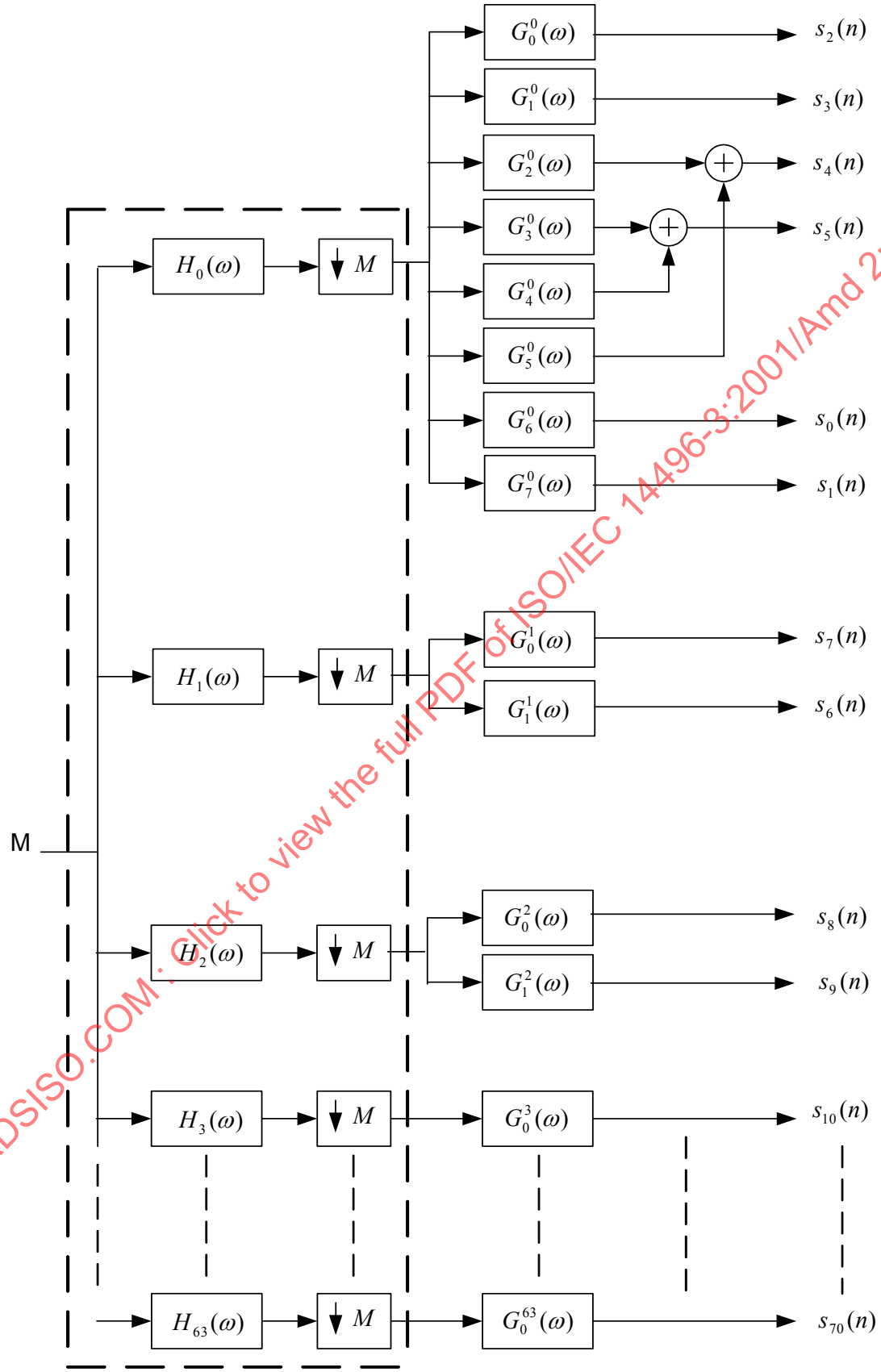

Figure 8.20 - Hybrid QMF analysis filterbank for the 10 and 20 stereo-bands configuration. The lower subbands of the 64 QMF (see dashed box) are further split to provide for increased resolution for the lower frequencies

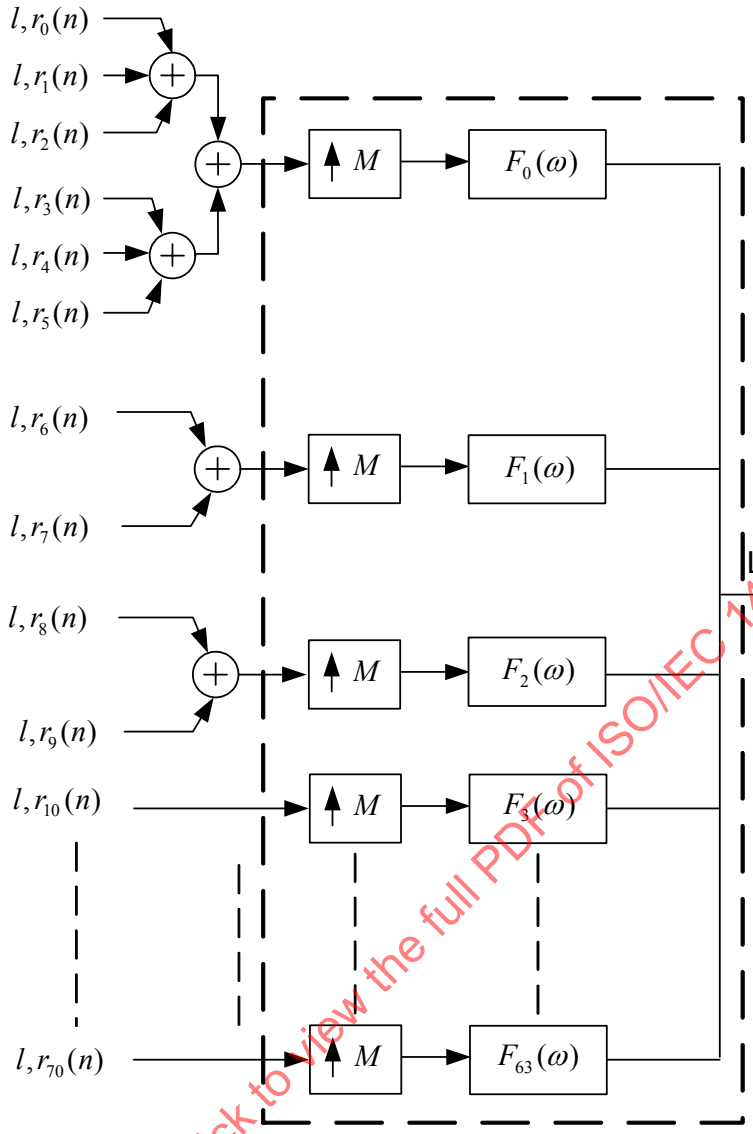

Figure 8.21 - Hybrid QMF synthesis filterbank for the 10 and 20 stereo-bands configuration. The coefficients offering higher resolution for the lower QMF channel are simply added prior to the synthesis with the 64 subbands QMF (see dashed box)

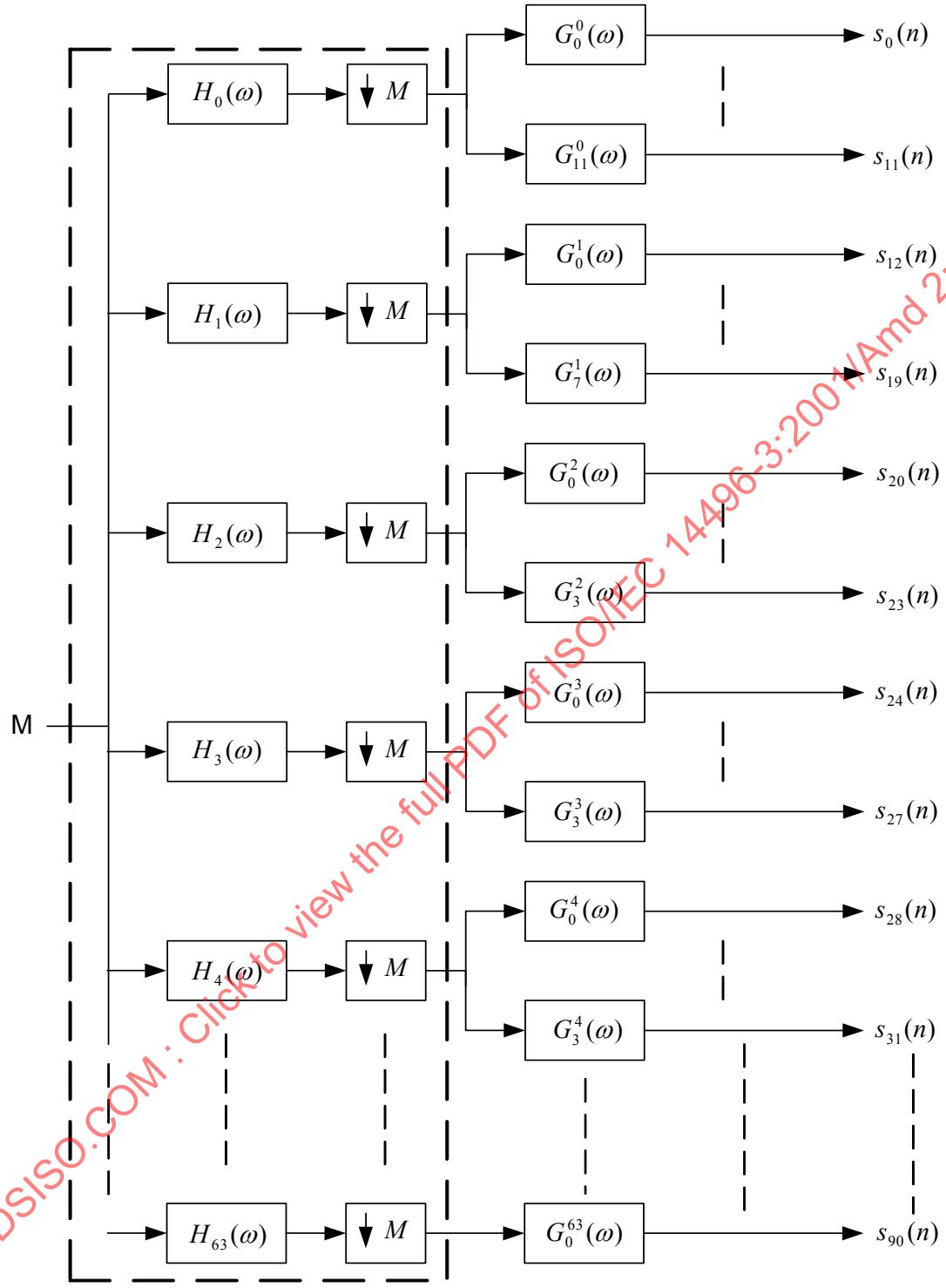

Figure 8.22 - Hybrid QMF analysis filterbank for the 34 stereo-bands configuration. The lower subbands of the 64 QMF (see dashed box) are further split to provide for increased resolution for the lower frequencies

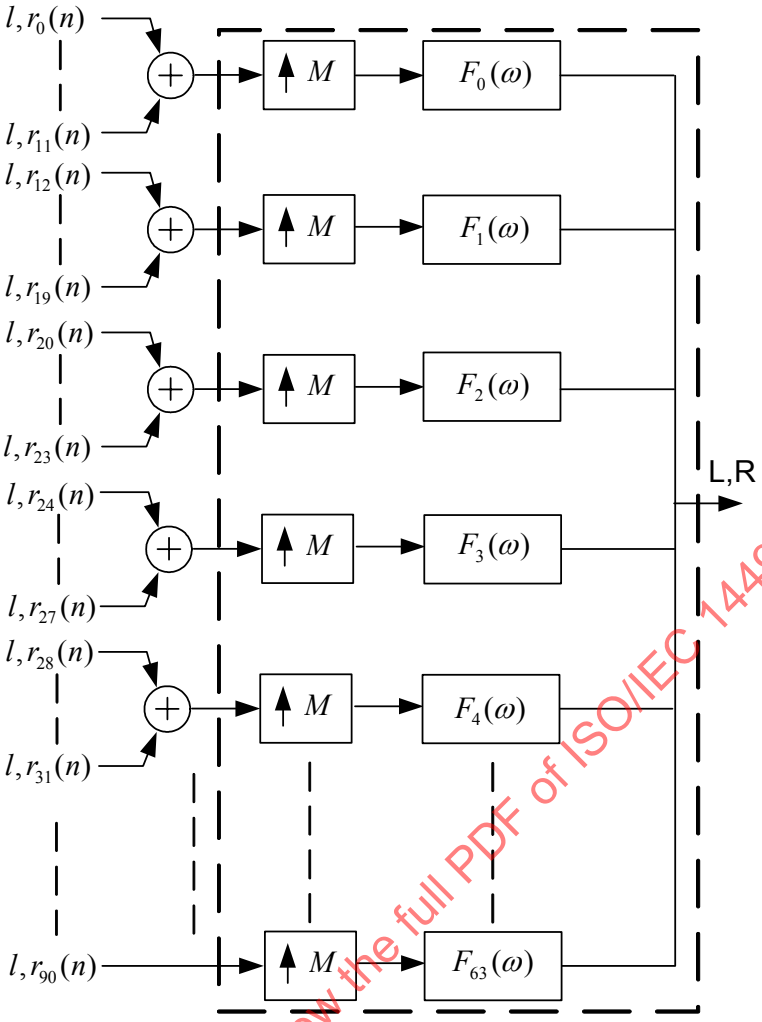

Figure 8.23 - Hybrid QMF synthesis filterbank for the 34 stereo-bands configuration. The coefficients offering higher resolution for the lower QMF subbands are simply added prior to the synthesis with the 64 subbands QMF (see dashed box)

In order to time align all the samples originating from the hybrid filterbank, the remaining QMF subbands that have not been filtered are delay compensated. This delay amounts to 6 QMF subband samples. This means $G_0^k(z) = z^{-6}$ for k=[3...63] (10,20 stereo bands) or k=[5...63] (34 stereo bands). In order to compensate for the overall delay of the hybrid analysis filterbank, the first 10 sets (6 from delay and 4 from QMF filter) of hybrid subbands are flushed and therefore not taken into account for processing.

The resultant of this operation is a slot of hybrid subband samples consisting of a LF (low frequency) sub QMF subband portion and HF (high frequency) QMF subband portion. As an illustration, see the hybrid slot in Figure 8.24.

8.6.4.4 Framing

One frame of parametric audio comprises two stereo frames of data. Stereo parameters within a stereo frame can be assigned to one or more slots. The stereo frame boundaries and the positions n_e of the slots that have been assigned stereo parameters to, define so called regions. Stereo parameters are defined for the last slot in a region. By means of these regions transients can be effectively handled. As illustrated by the bold solid lines in Figure 8.24, stereo parameters for non-assigned slots are obtained by means of interpolation. For the very first region ($region_0$), interpolation is performed with respect to the default parameters for index=0 (i.e.

IID=0, ICC=1 and IPD=OPD=0). Any existing slots after the last assigned slot in the stereo frame obtain the same stereo parameters from this last assigned slot ($region_2$). At stereo-frame borders ($region_3$), interpolation for the first region is performed with respect to the last slot in the previous frame. The stereo reconstruction is applied per region. For simplicity, the envelope index is only indicated when applicable.

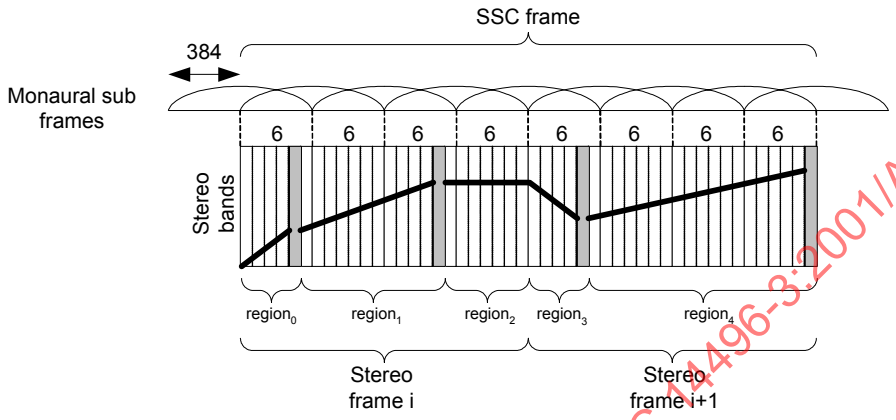

Figure 8.24 - One SSC frame comprises two stereo frames of data. The solid line illustrates the interpolation between stereo parameters for slots that have not been assigned stereo parameters to

8.6.4.5 De-correlation

By means of all-pass filtering and delaying, the sub subband samples $s_k(n)$ are converted into de-correlated sub subband samples $d_k(n)$, where k represents the frequency in the hybrid spectrum and n the time index.

8.6.4.5.1 Constants

F_s is the output sampling rate.

$DECAY\_SLOPE = 0.05$ is the all-pass filter decay slope.

$NR\_ALLPASS\_LINKS = 3$ is the number of filter links for the all-pass filter.

NR_PAR_BANDS is the number of frequency bands that can be addressed by the parameter index, b(k) (see Table 8.43 and Table 8.44).

$$NR\_PAR\_BANDS = \begin{cases} 20 & ,10 \text{ or } 20 \text{ stereo bands} \\ 34 & ,34 \text{ stereo bands} \end{cases}$$

NR_BANDS is the number of frequency bands that can be addressed by the sub subband index, k.

$$NR\_BANDS = \begin{cases} 71 & ,10 \text{ or } 20 \text{ stereo bands} \\ 91 & ,34 \text{ stereo bands} \end{cases}$$

$DECAY\_CUTOFF$ is the start frequency band for the all-pass filter decay slope.

$$DECAY\_CUTOFF = \begin{cases} 10 & ,10 \text{ or } 20 \text{ stereo bands} \\ 32 & ,34 \text{ stereo bands} \end{cases}$$

$NR\_ALLPASS\_BANDS$ is the number of all-pass filter bands.

$$NR\_ALLPASS\_BANDS = \begin{cases} 53 & ,F_s < 32kHz, 10 \text{ or } 20 \text{ stereo bands} \\ 73 & ,F_s < 32kHz, 34 \text{ stereo bands} \\ 30 & ,F_s \geq 32kHz, 10 \text{ or } 20 \text{ stereo bands} \\ 50 & ,F_s \geq 32kHz, 34 \text{ stereo bands} \end{cases}$$

$SHORT\_DELAY\_BAND$ is the first stereo band using the short, one sample delay.

$$SHORT\_DELAY\_BAND = \begin{cases} 71 & ,F_s < 32kHz, 10 \text{ or } 20 \text{ stereo bands} \\ 91 & ,F_s < 32kHz, 34 \text{ stereo bands} \\ 42 & ,F_s \geq 32kHz, 10 \text{ or } 20 \text{ stereo bands} \\ 62 & ,F_s \geq 32kHz, 34 \text{ stereo bands} \end{cases}$$

$$a_{Smooth} = \begin{cases} 0.6 & ,F_s < 32kHz \\ 0.25 & ,F_s \geq 32kHz \end{cases}$$ is the smoothing coefficient.

8.6.4.5.2 Calculate decorrelated signal, $d_k(z)$

The decorrelation process for the first $NR\_ALLPASS\_BANDS$ frequency bands of $s_k(n)$ is based on an all-pass filter described in the Z-domain below. Its transfer function for each band, k is defined by:

$$\mathbf{H}_k(z) = z^{-2} \cdot \varphi_{Fract}(k) \cdot \prod_{m=0}^{NR\_ALLPASS\_LINKS-1} \frac{\mathbf{Q}_{Fract\_allpass}(k,m) z^{-\mathbf{d}(m)} - \mathbf{a}(m)\mathbf{g}_{DecaySlope}(k)}{1 - \mathbf{a}(m)\mathbf{g}_{DecaySlope}(k)\mathbf{Q}_{Fract\_allpass}(k,m) z^{-\mathbf{d}(m)}} \quad ,$$

for $0 \leq k < NR\_ALLPASS\_BANDS$.

The fractional delay length matrix, $\mathbf{Q}_{Fract\_allpass}(k,m)$ and the fractional delay vector, $\varphi_{Fract}(k)$ are defined, by:

$$\mathbf{Q}_{Fract\_allpass}(k,m) = \exp\left(-i\pi\mathbf{q}(m)\mathbf{f}_{center}(k)\right) \quad ,\begin{cases} 0 \leq k < NR\_ALLPASS\_BANDS \\ 0 \leq m < NUM\_OF\_LINKS \end{cases},$$

and

$$\varphi_{Fract}(k) = \exp\left(-i\pi q_{\varphi}\mathbf{f}_{center}(k)\right) \quad ,0 \le k < NR\_ALLPASS\_BANDS\,,$$

where $i = \sqrt{-1}$ denotes the imaginary unit. For the frequency vector, $\mathbf{f}_{center}$, the fractional delay length vector, see Table 8.35 and Table 8.36. The vector $\mathbf{q}(k)$ is defined in Table 8.37. The fractional delay length constant, $q_{\varphi} = 0.39$.

For the filter coefficient vector $\mathbf{a}(m)$ and the delay length vector $\mathbf{d}(m)$ see Table 8.34.

The vector $\mathbf{g}_{DecaySlope}$ contains time invariant factors for making the all-pass filter frequency variant. It is defined by:

$$\mathbf{g}_{DecaySlope}(k) = \begin{cases} \max\left\langle 0, 1 - DECAY\_SLOPE \cdot \left(k - DECAY\_CUTOFF\right)\right\rangle & ,k > DECAY\_CUTOFF \\ 1 & ,otherwise \end{cases}$$

for $0 \le k < NR\_ALLPASS\_BANDS$.

For the upper bands, i.e. for $NR\_ALLPASS\_BANDS \le k < NR\_BANDS$ the transfer function, $\mathbf{H}_k(z)$ equals a delay according to:

$\mathbf{H}_k(z) = z^{-D(k)}$, where $D(k)$ is defined by:

$$D(k) = \begin{cases} 14 & ,NR\_ALLPASS\_BANDS \le k < SHORT\_DELAY\_BAND \\ 1 & ,SHORT\_DELAY\_BAND \le k < NR\_BANDS \end{cases}.$$

8.6.4.5.3 Perform transient detection

To be able to handle transients and other fast time-envelopes, the all-pass filter has to be attenuated at those signals. It is done by the following scheme:

First define the input power matrix, $\mathbf{P}(i,n)$ that contains the sum of the squared sub subband samples of each parameter band. As indicated in Figure 8.25, *n* runs from n_0 to n_{L-1}.

$$\mathbf{P}(i,n) = \sum_{i=b(k)} \left|s_k(n)\right|^2 \quad ,0 \le i < NR\_PAR\_BANDS\,,$$

where b(k) is defined in Table 8.43 and Table 8.44. Apply peak decay on the input power signal according to:

$$\mathrm{P}_{PeakDecayNrg}(i,n) = \begin{cases} \mathrm{P}(i,n) & ,\alpha\mathrm{P}_{PeakDecayNrg}(i,n-1) < \mathrm{P}(i,n) \\ \alpha\mathrm{P}_{PeakDecayNrg}(i,n-1) & ,otherwise \end{cases}.$$

for $0 \le i < NR\_PAR\_BANDS$. α is the peak decay factor defined in Table 8.38.

Subsequently, filter the input power and peak decay power signals with the Z-domain transfer function,

$H_{Smooth}(z)$:

$$\mathbf{P}_{SmoothNrg}(i,z) = H_{Smooth}(z) \cdot \mathbf{P}(i,z),$$

$$\mathbf{P}_{SmoothPeakDecayDiffNrg}(i,z) = H_{Smooth}(z) \cdot \left(\mathbf{P}_{PeakDecayNrg}(i,z) - \mathbf{P}(i,z)\right),$$

for $0 \le i < NR\_PAR\_BANDS$, where

$$H_{Smooth}(z) = \frac{a_{Smooth}}{1 + (a_{Smooth} - 1) \cdot z^{-1}}.$$

The transient attenuator, $\mathbf{G}_{TransientRatio}$ is then calculated as follows:

$$\mathbf{G}_{TransientRatio}(i,n) = \begin{cases} \dfrac{\mathbf{P}(i,n)}{\gamma \cdot \mathbf{P}_{SmoothPeakDecayDiffNrg}(i,n)} & , \gamma \cdot \mathbf{P}_{SmoothPeakDecayDiffNrg}(i,n) > \mathbf{P}(i,n) \\ 1 & , otherwise \end{cases},$$

for $0 \le i < NR\_PAR\_BANDS$, where $\gamma = 1.5$ is a transient impact factor.

Finally map the transient attenuator, $\mathbf{G}_{TransientRatio}$ to bands according to:

$$\mathbf{G}_{TransientRatioMapped}(k,n) = \mathbf{G}_{TransientRatio}(b(k),n) \quad , 0 \le k < NR\_BANDS.$$

8.6.4.5.4 Apply transient reduction to decorrelated signal

Let $d_k(z)$ be the decorrelated signal and $s_k(z)$ the mono input signal in the Z-domain for each band. Then $d_k(z)$ is defined according to:

$d_k(z) = \mathbf{G}_{TransientRatioMapped}(k,z) \cdot \mathbf{H}_k(z) \cdot s_k(z)$, where $0 \le k < NR\_BANDS$.

Table 8.34 - Filter coefficient vector, delay length vectors $\mathbf{d}_{24kHz}(m)$ and $\mathbf{d}_{48kHz}(m)$.

m	$\mathbf{a}(m)$	$\mathbf{d}_{24kHz}(m)$	$\mathbf{d}_{48kHz}(m)$
0	0.65143905753106	1	3
1	0.56471812200776	2	4
2	0.48954165955695	3	5

Delay length vector, $\mathbf{d} = \begin{cases} \mathbf{d}_{24kHz} & , F_s < 32kHz \\ \mathbf{d}_{48kHz} & , F_s \ge 32kHz \end{cases}$.

Table 8.35 - Delay length vector $\mathbf{f}_{center\_20}$.

k	$\mathbf{f}_{center\_20}(k)$	k	$\mathbf{f}_{center\_20}(k)$
0	-3/8	5	7/8
1	-1/8	6	2.5/2
2	1/8	7	3.5/2
3	3/8	8	4.5/2
4	5/8	9	5.5/2

Table 8.36 - Delay length vector $\mathbf{f}_{center\_34}$.

k	$\mathbf{f}_{center\_34}(k)$	k	$\mathbf{f}_{center\_34}(k)$
0	1/12	16	9/8
1	3/12	17	11/8
2	5/12	18	13/8
3	7/12	19	15/8
4	9/12	20	9/4
5	11/12	21	11/4
6	13/12	22	13/4
7	15/12	23	7/4
8	17/12	24	17/4
9	-5/12	25	11/4
10	-3/12	26	13/4
11	-1/12	27	15/4
12	17/8	28	17/4
13	19/8	29	19/4
14	5/8	30	21/4
15	7/8	31	15/4

$$\mathbf{f}_{center\_20}(k) = k + \frac{1}{2} - 7 \quad , 10 \le k < NR\_ALLPASS\_BANDS \, ,$$

$$\mathbf{f}_{center\_34}(k) = k + \frac{1}{2} - 27 \quad , 32 \le k < NR\_ALLPASS\_BANDS \, ,$$

$$\mathbf{f}_{center} = \begin{cases} \mathbf{f}_{center\_20} & , NR\_PAR\_BANDS = 20 \\ \mathbf{f}_{center\_34} & , NR\_PAR\_BANDS = 34 \end{cases} .$$

Table 8.37 - Fractional delay length vector $\mathbf{q}(m)$.

m	$\mathbf{q}(m)$
0	0.43
1	0.75
2	0.347

Table 8.38 - Peak Decay Factors $\alpha_{Decay24kHz}$ and $\alpha_{Decay48kHz}$.

$\alpha_{Decay24kHz}$	$\alpha_{Decay48kHz}$
0.58664621951003	0.76592833836465

Peak decay factor, $\alpha = \begin{cases} \alpha_{Decay24kHz} & , F_s < 32kHz \\ \alpha_{Decay48kHz} & , F_s \geq 32kHz \end{cases}$.

8.6.4.6 Stereo Processing

The sets of sub subband samples $s_k(n)$ and $d_k(n)$ are now processed according to the stereo cues. These cues are defined per stereo band. All the hybrid subband samples within a stereo band are processed according to the cues in that respective stereo band. Table 8.43 and Table 8.44 indicate the hybrid subband samples that fall into each stereoband for the (10,20) and 34 stereoband configuration. *k* traverses the range from [0…70] or [0…90] for the (10,20) or 34 stereoband configuration respectively.

8.6.4.6.1 Mapping

The number of stereo bands that is actually used for the processing of the cues depends on the number of available parameters for IID and ICC according to the relation given in Table 8.39.

Table 8.39 - The number of stereo bands depends on the number of parameters for IID and ICC.

Number of IID parameters	number of ICC parameters	number of stereo bands
10	10	20 (i.e., 10,20 stereo band configuration)
10	20	
20	10	
20	20	
10,20	34	34
34	10,20	

In the case the number of parameters for IID and ICC differs from the number of stereo bands, as dictated in Table 8.39, a mapping from the lower number to the higher number of parameters is required. For the mapping from 10 to 20 parameters this is realised by duplicating every parameter as shown in Table 8.40. For the mapping from 20 to 34 parameters this is realized according to Table 8.40. For the mapping from 10 to 34 parameters, the 10 parameters are first mapped to 20 parameters and subsequently to 34 parameters. Table 8.41 provides the inverse mapping from 34 to 20 parameters.

Table 8.40 - Mapping from 10 to 20 to 34 parameters.

parameter grid			parameter grid		
34	20	10	34	20	10
idx_0	idx_0	idx_0	idx_{17}	idx_{11}	idx_5
idx_1	$(idx_0+idx_1)/2$	idx_0	idx_{18}	idx_{12}	idx_6
idx_2	idx_1	idx_0	idx_{19}	idx_{13}	idx_6
idx_3	idx_2	idx_1	idx_{20}	idx_{14}	idx_7
idx_4	$(idx_2+idx_3)/2$	idx_1	idx_{21}	idx_{14}	idx_7
idx_5	idx_3	idx_1	idx_{22}	idx_{15}	idx_7
idx_6	idx_4	idx_2	idx_{23}	idx_{15}	idx_7
idx_7	idx_4	idx_2	idx_{24}	idx_{16}	idx_8
idx_8	idx_5	idx_2	idx_{25}	idx_{16}	idx_8
idx_9	idx_5	idx_2	idx_{26}	idx_{17}	idx_8
idx_{10}	idx_6	idx_3	idx_{27}	idx_{17}	idx_8
idx_{11}	idx_7	idx_3	idx_{28}	idx_{18}	idx_9
idx_{12}	idx_8	idx_4	idx_{29}	idx_{18}	idx_9
idx_{13}	idx_8	idx_4	idx_{30}	idx_{18}	idx_9
idx_{14}	idx_9	idx_4	idx_{31}	idx_{18}	idx_9
idx_{15}	idx_9	idx_4	idx_{32}	idx_{19}	idx_9
idx_{16}	idx_{10}	idx_5	idx_{33}	idx_{19}	idx_9

Table 8.41 – Mapping of IID, ICC, IPD and OPD parameters from 34 stereo bands to 20 stereo bands. For IPD and OPD parameters, this mapping applies up to and including idx_{10} and idx_{16} for 20 and 34 stereo bands respectively

20 stereo bands	34 stereo bands
idx_0	$(2*idx_0+idx_1)/3$
idx_1	$(idx_1+2*idx_2)/3$
idx_2	$(2*idx_3+idx_4)/3$
idx_3	$(idx_4+2*idx_5)/3$
idx_4	$(idx_6+idx_7)/2$
idx_5	$(idx_8+idx_9)/2$
idx_6	idx_{10}
idx_7	idx_{11}
idx_8	$(idx_{12}+idx_{13})/2$
idx_9	$(idx_{14}+idx_{15})/2$
idx_{10}	idx_{16}
idx_{11}	idx_{17}
idx_{12}	idx_{18}
idx_{13}	idx_{19}
idx_{14}	$(idx_{20}+idx_{21})/2$
idx_{15}	$(idx_{22}+idx_{23})/2$
idx_{16}	$(idx_{24}+idx_{25})/2$
idx_{17}	$(idx_{26}+idx_{27})/2$
idx_{18}	$(idx_{28}+idx_{29}+idx_{30}+idx_{31})/4$
idx_{19}	$(idx_{32}+idx_{33})/2$

The averaging process denoted by e.g. $(2*idx_0 + idx_1)/2$ in Table 8.40 and Table 8.41 is carried out for the integer index representation idx_k of the IID or ICC parameters prior to dequantization, according to ANSI-C integer arithmetic.

The IPD/OPD parameters follow the mapping for the IID parameters, taking into account the relative amount of parameters for IPD/OPD as given by Table 8.19. Consequently the same mapping as for IID is applied, but only for a lower number of parameters for IPD/OPD. For the upper stereo bands, where no IPD/OPD data is transmitted, the IPD/OPD parameters are set to zero.

If the number of stereo bands changes from 10,20 in the previous frame to 34 in the current frame, the stereo parameters from the previous frame are mapped to 34 stereo bands according to Table 8.40 prior to further processing of the current. The frequency resolution of the hybrid QMF analysis filterbank (see subclause 8.6.4.3) is changed instantaneously to the 34 stereo band configuration. The state variable of the decorrelation process are mapped from *NR_BANDS*==71 to *NR_BANDS*==91 configuration according to Table 8.42. The state variables for sub subbands in *NR_BANDS*==91 configuration not listed in Table 8.42 are reset to zero.

If the number of stereo bands changes from 34 in the previous frame to 10,20 in the current frame, the stereo parameters from the previous frame are mapped to 20 stereo bands according to Table 8.41 prior to further processing of the current. The frequency resolution of the hybrid QMF analysis filterbank (see subclause 8.6.4.3) is changed instantaneously to the 20 stereo band configuration. The state variable of the decorrelation process are mapped from *NR_BANDS*==91 to *NR_BANDS*==71 configuration according to Table 8.42. The state variables for sub subbands in *NR_BANDS*==91 configuration not listed in Table 8.42 are discarded.

Table 8.42 - Mapping of state variables of decorrelation process between 10,20 and 34 stereo band configurations

NR_BANDS==91	*NR_BANDS*==71	
Sub subband index *k*	Sub subband index *k*	
9	0	Sub QMF
11	1	
0	2	
2	3	
3	4	
5	5	
16	6	
18	7	
20	8	
21	9	
26	10	
28	11	
32	12	QMF (only)
[33, 89]	[13, 69]	
90	70	

8.6.4.6.2 Mixing

In order to generate the QMF subband signals for the subband samples $n = n_e + 1 \ldots n_{e+1}$ the parameters at position n_e and n_{e+1} are required as well as the subband domain signals $s_k(n)$ and $d_k(n)$ for $n = n_e + 1 \ldots n_{e+1}$ (see Figure 8.25). For IPD/OPD, the parameters at position n_{e-1} are needed in addition. n_e represents the start position for envelope e. In the case frameclass == %1 (VAR_BORDERS), the border positions n_e are obtained by borderposition[e]. In the case frameclass == %0 (FIX_BORDERS), the border positions n_e are obtained by means of

$$n_e = \left\lfloor \frac{numQMFSlots * (e+1)}{num\_env} \right\rfloor - 1, \qquad e = [0, \ldots, num\_env - 1].$$

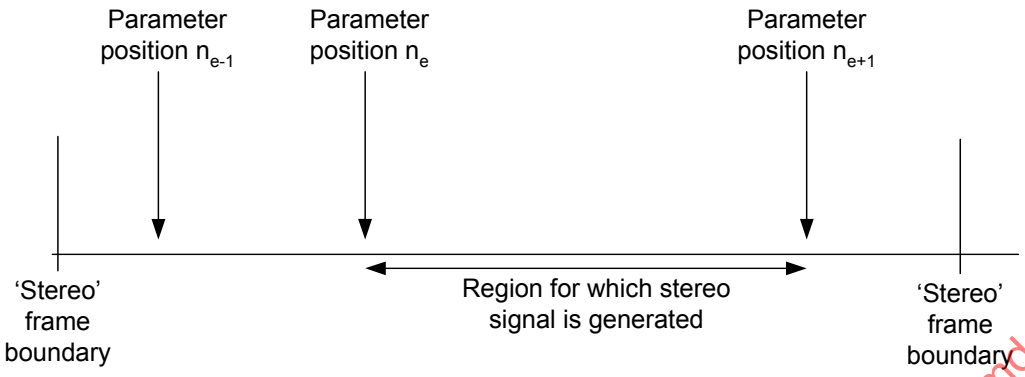

Figure 8.25 - Region for which the stereo subband signals are generated

The stereo sub subband signals are constructed as:

$$l_k(n) = H_{11}(k,n)s_k(n) + H_{21}(k,n)d_k(n)$$
$$r_k(n) = H_{12}(k,n)s_k(n) + H_{22}(k,n)d_k(n)$$

In order to obtain the matrices $H_{11}(k,n)$, $H_{12}(k,n)$, $H_{21}(k,n)$, $H_{22}(k,n)$, the vectors $h_{11}(b)$, $h_{12}(b)$, $h_{21}(b)$, $h_{22}(b)$ need to be calculated first, where the parameter b is used as parameter index. First for the parameter position n_{e+1} the intensity differences (IID) are transformed to the linear domain.

$$c(b) = 10^{\frac{iid(b)}{20}},$$

where *iid*(*b*) represents the decoded IID value for stereo band *b* in dB. Depending on the ICC mode configuration, either mixing procedure R_a or R_b is used, see Table 8.22. For both mixing procedures, the parameters for parameter position n_{e+1} are used.

8.6.4.6.2.1 Mixing procedure R_a

In the case mixing procedure R_a is used the following method is applied.

From the intensity differences two scale-factor vectors c_1 and c_2 are calculated.

$$c_1(b) = \frac{\sqrt{2}}{\sqrt{1+c^2(b)}}$$
$$c_2(b) = \frac{\sqrt{2}c(b)}{\sqrt{1+c^2(b)}}$$

From these and the ICC parameter $\rho(b)$, the coefficients $h_{xy}(b)$ are calculated according to

$$\alpha(b) = \frac{1}{2}\arccos(\rho(b))$$
$$\beta(b) = \alpha(b)\frac{c_1(b) - c_2(b)}{\sqrt{2}}$$
$$h_{11}(b) = \cos(\alpha(b) + \beta(b))c_2(b)$$
$$h_{12}(b) = \cos(\beta(b) - \alpha(b))c_1(b)$$
$$h_{21}(b) = \sin(\alpha(b) + \beta(b))c_2(b)$$
$$h_{22}(b) = \sin(\beta(b) - \alpha(b))c_1(b)$$

8.6.4.6.2.2 Mixing procedure R_b

In the case mixing procedure R_b is used the following method is applied.

In order to prevent instabilities, in the case the value of $\rho(b)$ is smaller than 0.05, $\rho(b)$ is set to 0.05. In the case $c(b)$ is not equal to 1

$$\alpha(b) = \frac{1}{2}\arctan\left(\frac{2c(b)\rho(b)}{c^2(b) - 1}\right),$$

otherwise $\alpha(b) = \frac{\pi}{4}$. After modulo correction of $\alpha(b)$, again the value of $c(b)$ and $\rho(b)$ are used to derive the coefficients $h_{xy}(b)$.

$$\alpha(b) = \alpha(b) - \left\lfloor \frac{\alpha(b)}{\frac{1}{2}\pi} \right\rfloor \frac{1}{2}\pi,$$
$$\mu(b) = 1 + \frac{4\rho^2(b) - 4}{(c(b) + c^{-1}(b))^2},$$
$$\gamma(b) = \arctan\left(\sqrt{\frac{1 - \sqrt{\mu(b)}}{1 + \sqrt{\mu(b)}}}\right),$$
$$h_{11}(b) = \sqrt{2}\cos(\alpha(b))\cos(\gamma(b)),\ .$$
$$h_{12}(b) = \sqrt{2}\sin(\alpha(b))\cos(\gamma(b)),$$
$$h_{21}(b) = -\sqrt{2}\sin(\alpha(b))\sin(\gamma(b)),$$
$$h_{22}(b) = \sqrt{2}\cos(\alpha(b))\sin(\gamma(b)).$$

8.6.4.6.3 Phase parameters

8.6.4.6.3.1 Phase parameters disabled

In the case IPD and OPD are disabled as indicated by (**enable_ipdopd**==0) the following procedure is applied. In order to obtain $H_{11}(k, n_{e+1})$, $H_{12}(k, n_{e+1})$, $H_{21}(k, n_{e+1})$ and $H_{22}(k, n_{e+1})$ we use the following equations

$$H_{11}(k,n_{e+1}) = h_{11}(b(k))$$
$$H_{12}(k,n_{e+1}) = h_{12}(b(k))$$
$$H_{21}(k,n_{e+1}) = h_{21}(b(k))$$
$$H_{22}(k,n_{e+1}) = h_{22}(b(k))$$

where *b*(*k*) is defined in Table 8.43 and Table 8.44.

8.6.4.6.3.2 Phase parameters enabled

In the case IPD and OPD are enabled as indicated by (enable_ipdopd==1) the following procedure is applied. First the IPD and OPD values are smoothed over time according to

$$\varphi_{opd}(b) = \angle\left\{\frac{1}{4}\exp(j\cdot opd(b,n_{e-1})) + \frac{1}{2}\exp(j\cdot opd(b,n_e)) + \exp(j\cdot opd(b,n_{e+1}))\right\}$$
$$\varphi_{ipd}(b) = \angle\left\{\frac{1}{4}\exp(j\cdot ipd(b,n_{e-1})) + \frac{1}{2}\exp(j\cdot ipd(b,n_e)) + \exp(j\cdot ipd(b,n_{e+1}))\right\}$$

In the case the number of IPD/OPD parameters for parameter position n_{e-1} and/or n_e are different from the number of IPD/OPD parameters for parameter position n_{e+1}, these are mapped to the number of IPD/OPD parameters for parameter position n_{e+1} using Table 8.40 and Table 8.41.

$$\varphi_1(b) = \varphi_{opd}(b)$$
$$\varphi_2(b) = \varphi_{opd}(b) - \varphi_{ipd}(b)$$

Finally, in order to get to $H_{11}(k,n_{e+1})$, $H_{12}(k,n_{e+1})$, $H_{21}(k,n_{e+1})$ and $H_{22}(k,n_{e+1})$, the following equations are applied.

$$H_{11}(k,n_{e+1}) = h_{11}(b(k))\cdot\exp(j\varphi_1(b(k)))$$
$$H_{12}(k,n_{e+1}) = h_{12}(b(k))\cdot\exp(j\varphi_2(b(k)))$$
$$H_{21}(k,n_{e+1}) = h_{21}(b(k))\cdot\exp(j\varphi_1(b(k)))$$
$$H_{22}(k,n_{e+1}) = h_{22}(b(k))\cdot\exp(j\varphi_2(b(k)))$$

where Table 8.43 and Table 8.44 are used to translate from parameter indexing to subband indexing. For indices denoted with a *, we use:

$$H_{11}(k,n_{e+1}) = h_{11}(b(k))\cdot\exp(-j\varphi_1(b(k)))$$
$$H_{12}(k,n_{e+1}) = h_{12}(b(k))\cdot\exp(-j\varphi_2(b(k)))$$
$$H_{21}(k,n_{e+1}) = h_{21}(b(k))\cdot\exp(-j\varphi_1(b(k)))$$
$$H_{22}(k,n_{e+1}) = h_{22}(b(k))\cdot\exp(-j\varphi_2(b(k)))$$

Table 8.43 - Mapping of parameters from 20 bands to 71 sub subbands.

sub subband index *k*	QMF channel	Parameter index *b*(*k*)	
0	0	1*	Sub QMF
1	0	0*	
2	0	0	
3	0	1	
4	0	2	
5	0	3	
6	1	4	
7	1	5	
8	2	6	
9	2	7	
10	3	8	QMF (only)
11	4	9	
12	5	10	
13	6	11	
14	7	12	
15	8	13	
16-17	9-10	14	
18-20	11-13	15	
21-24	14-17	16	
25-29	18-22	17	
30-41	23-34	18	
42-70	35-63	19	

Table 8.44 - Mapping of parameters from 34 bands to 91 sub subbands.

Sub subband index *k*	QMF channel	Parameter index *b*(*k*)	
0	0	0	Sub QMF
1	0	1	
2	0	2	
3	0	3	
4	0	4	
5	0	5	
6-7	0	6	
8	0	7	
9	0	2*	
10	0	1*	
11	0	0*	
12-13	1	10	
14	1	4	
15	1	5	
16	1	6	
17	1	7	
18	1	8	
19	1	9	
20	2	10	
21	2	11	
22	2	12	
23	2	9	
24	3	14	
25	3	11	
26	3	12	
27	3	13	
28	4	14	
29	4	15	
30	4	16	
31	4	13	

32	5	16	QMF (only)
33	6	17	
34	7	18	
35	8	19	
36	9	20	
37	10	21	
38-39	11-12	22	
40-41	13-14	23	
42-43	15-16	24	
44-45	17-18	25	
46-47	19-20	26	
48-50	21-23	27	
51-53	24-26	28	
54-56	27-29	29	
57-59	30-32	30	
60-63	33-36	31	
64-67	37-40	32	
68-90	41-63	33	

8.6.4.6.4 Interpolation

The intermediate values for $H_{11}(k,n)$, $H_{12}(k,n)$, $H_{21}(k,n)$ and $H_{22}(k,n)$ at positions $n = n_e + 1 \ldots n_{e+1}$ are obtained by means of linear interpolation conforming to.

$$H_{11}(k,n) = H_{11}(k,n_e) + (n-n_e)\frac{H_{11}(k,n_{e+1}) - H_{11}(k,n_e)}{n_{e+1} - n_e}$$

$$H_{12}(k,n) = H_{12}(k,n_e) + (n-n_e)\frac{H_{12}(k,n_{e+1}) - H_{12}(k,n_e)}{n_{e+1} - n_e}$$

$$H_{21}(k,n) = H_{21}(k,n_e) + (n-n_e)\frac{H_{21}(k,n_{e+1}) - H_{21}(k,n_e)}{n_{e+1} - n_e}$$

$$H_{22}(k,n) = H_{22}(k,n_e) + (n-n_e)\frac{H_{22}(k,n_{e+1}) - H_{22}(k,n_e)}{n_{e+1} - n_e}$$

8.6.4.7 Hybrid QMF synthesis filterbank

The stereo processed hybrid subband signals $l_k(n)$ and $r_k(n)$ are fed into the hybrid synthesis filterbanks, which are implemented as adders of sub QMF samples. This is illustrated in Figure 8.21 and Figure 8.23. The two synthesis filterbanks are identical to the 64 complex QMF synthesis filterbank as defined in ISO/IEC 14496-3/AMD1:2003 subclause 4.6.18.4.2. The input to the filterbank are slots of 64 QMF samples. For each slot the filterbank outputs one block of 64 samples of the one channel of the reconstructed stereo signal. There are two independent instances of the QMF synthesis filterbank for the left and right channel, respectively.

8.6.5 Start/stop situations for decoding

Decoding of an excerpt has to be started and ended in a certain way. This section explains how to deal with start and end of the decoding process.

8.6.5.1 Start decoding

The start of decoding occurs for the first frame of an excerpt, or during random access in an excerpt.

No special actions need to be taken for transients.

For sinusoids a previous (non-existent) sub-frame must be filled with a zero signal. The overlap-add method then generates a natural fade-in for the sinusoidal components of the first sub-frame.

For noise a previous (non-existent) sub-frame must be filled with a zero signal. The overlap-add method then generates a natural fade-in for the noise component of the first sub-frame.

No special actions are required for parametric stereo.

8.6.5.2 Stop decoding

The stop of decoding occurs for the last frame of an excerpt, or during random access in an excerpt (stop decoding process “manually” (e.g. stop, skip, pause)).

For a step transient no special precautions are required. For a Meixner transient it is possible that the tail has not ended at the end of the excerpt. It is advised to stop generating output for the Meixner transient at the end of the excerpt.

For sinusoids a next (non-existent) sub-frame must be filled with a zero signal. The overlap-add method then generates a natural fade-out for the sinusoidal components of the last sub-frame.

For noise a next (non-existing) sub-frame must be filled with a zero signal. The overlap-add method then generates a natural fade-out for the noise component of the last sub-frame.

For parametric stereo no special actions are required.

8.7 References

[FFT] *High-Precision Fourier Analysis of Sounds Using Signal Derivatives.* M. Desainte-Catherine and S. Marchand, Journal of the Audio Engineering Society, Vol. 48, No. 7/8, 2000 July/August.

[Laguerre1] A.C. den Brinker. Stability of linear predictive structures using IIR filters. In *Proc. 12th ProRISC* Workshop, pages 317–320, Veldhoven (NL), 29-30 Nov. 2001.

[Laguerre2] V. Voitishchuk, A.C. den Brinker, and S.J.L. van Eijndhoven. Alternatives for warped linear predictors. In *Proc. 12th ProRISC Workshop*, pages 710–713, Veld-hoven (NL), 29-30 Nov. 2001.

[HILN] ISO/IEC 14496-3 Version 2 HILN reference code.

[Rothweiler] *A Rootfinding algorithm for line spectral frequencies*. J. Rothweiler. ICASSP March 1991.

[Breebaart] Binaural processing model based on contralateral inhibition I. Model setup, J. Breebaart, S. van de Par and A. Kohlrausch, J. Acoust. Soc. Am., 110:1074–1088, 2001.

Annex A
(normative)
Combination of the SBR tool with the parametric stereo tool

8.A.1 Overview

The parametric stereo coding tool (PS tool) can be used in combination with the SBR tool as defined in subclause 4.6.18. In this case, a 1-channel audio signal is conveyed by AAC+SBR (i.e., HE-AAC) and the PS tool is used to reconstruct a 2-channel stereo signal from this monaural signal. The bitstream element ps_data() as defined in subclause 8.4.2 conveys the information needed by the PS tool and is carried in the sbr_extension() container of the SBR bitstream.

The usage of this parametric stereo extension to HE-AAC is signalled implicitly in the bitstream. Hence, if an sbr_extension() with bs_extension_id==EXTENSION_ID_PS is found in the SBR part of the bitstream, a decoder supporting the combination of SBR and PS shall operate the PS tool to generate a stereo output signal. If no ps_data() element is available in the SBR part of a monaural HE-AAC bitstream, the normal monaural signal is generated by the SBR tool.

8.A.2 Bitstream syntax and semantics

The bitstream element ps_data() as defined in subclause 8.4.2 is carried in the sbr_extension() container (see Table 8.A.1 below) provided by the SBR bitstream defined in subclause 4.4.2.8. The semantics of the bs_extension_id field are given in Table 8.A.2, which replaces Table 4.97 “bs_extension_id.”

Table 8.A.1 – Syntax of sbr_extension()

Syntax	No. of bits	Mnemonic

sbr_extension(bs_extension_id, num_bits_left)
{
  switch (bs_extension_id) {
  case EXTENSION_ID_PS:
  num_bits_left -= ps_data();
  break;
  default:
  bs_fill_bits;
  num_bits_left = 0;
  break;
  }
}
	num_bits_left	Note 1
		
		bslbf

Note 1: ps_data() returns the number of bits read.		

Table 8.A.2 – Values of the bs_extension_id field

Symbol	Value	Purpose
EXTENSION_ID_PS	2	Parametric Stereo Coding
	all other values	reserved

8.A.3 Decoding process

Semantics and decoding process for the PS tool are defined in subclauses 8.5.2 and 8.6.4, respectively. When the PS tool is combined with SBR, a stereo frame is identical to an SBR frame and consists of 32 complex samples per QMF band for 1024 framing of the AAC (30 samples for 960 framing). The initial 64-band QMF analysis filterbank of the PS tool is removed and the 64-band QMF representation of the monaural signal generated by the SBR tool as available directly prior to SBR's 64-band QMF synthesis filterbank is used

as input to the PS tool. The monaural 64-band QMF synthesis filterbank of the SBR tool is obsolete and hence removed. Instead, the two 64-band QMF synthesis filterbanks at the output of the PS tool are used to generate the stereo audio signal.

As shown in Figure 4.46, "Synchronization and timing" in the SBR tool description, there are 6 QMF samples look-ahead available in the low-band buffer, i.e., $\mathbf{X}_{Low}(k,l)$ with 34<=*l*<40 for 1024 framing. Hence, the hybrid filter structure of the PS tool, where the lowest 3 or 5 QMF bands (all of which are in the low-band) are split up further with help of 13-tap linear-phase FIR filters, does not introduce any algorithmic delay when the PS tool is inserted between SBR processing and the final 64-band QMF synthesis filterbanks. This means that the delay as depicted in Figure 8.19 is obsolete and thus removed.

Hence, the input to the PS tool is a matrix $\mathbf{X}_{input}(k,l)$ defined according to:

$$\mathbf{X}_{input}(k,l) = \begin{cases} \mathbf{X}_{Low}\left(k, l + t_{HFAdj}\right) & , 0 \le k < 5, numTimeSlots \cdot RATE \le l < numTimeSlots \cdot RATE + 6 \\ \mathbf{X}(k,l) & , 0 \le k < 64, 0 \le l < numTimeSlots \cdot RATE \end{cases}$$

where $\mathbf{X}_{Low}(k,l)$, $\mathbf{X}(k,l)$, t_{HFAdj} , *numTimeSlots*, and *RATE* are defined in subclause 4.6.18 SBR Tool. Furthermore, *numQMFSlots=numTimeSlots*RATE.*

The PS tool uses a complex-valued QMF representation and therefore cannot be used in combination with the low power version of the SBR tool. If DRC is used in combination with SBR as defined in subclause 4.5.2.7.5, DRC is applied in the QMF domain to the output of the PS tool immediately prior to the QMF synthesis filterbanks. The same *factor(k,l)* is applied to both the left and the right audio channel.

8.A.4 Baseline version of the parametric stereo coding tool

In order to facilitate implementation of the PS decoder tool on platforms with very limited computational resources, a baseline version of the PS tool is defined. A PS decoder implementing this baseline version always uses the hybrid filter structure for 20 stereo bands and does not implement IPD/OPD synthesis. This results in a reduction of the computational complexity by approximately 25% when compared to unrestricted PS tool. The baseline version of the PS tool supports the complete bitstream syntax for ps_data(). However, IPD/OPD data is ignored and reset to IPD=OPD=0 prior to stereo synthesis. If a 34 stereo band configuration is used for IID or ICC parameters in the bitstream, the decoded parameters are mapped to 20 stereo bands according to Table 8.41. The averaging process denoted by e.g. $(2*idx_0+idx_1)/3$ in this table is carried out according to ANSI-C integer arithmetic for the integer index representation idx_k of the IID or ICC parameters prior to dequantization.

Annex B
(normative)
Normative Tables

8.B.1 Huffman tables for SSC

The function ssc_huff_dec() is used as:

data = ssc_huff_dec (t_huff, codeword),

where *t_huff* is the selected Huffman table and *codeword* is the word read from the bitstream. The return value *data*, is a Huffman table index corresponding to a specific code word.

Huffman table overview:

Table 8.B.1 – huff_sgrid

Index	huff_sgrid
0	100001
1	11101
2	11110
3	1100
4	1101
5	1010
6	0111
7	001
8	1011
9	0110
10	1001
11	0101
12	0000
13	0001
14	11100
15	01001
16	111111
17	111110
18	100000
19	010001
20	010000
21	10001

Table 8.B.2 – huff_sampba

index	huff_sampba	index	huff_sampba	index	huff_sampba	index	huff_sampba
0	110010010	64	1101	128	0110	192	110010011001
8	0100111	72	001	136	11000	200	110010011000100
16	1100101	80	000	144	01000	208	110010011000101
24	110011	88	1111	152	010010	216	110010011000110
32	01110	96	1110	160	0100110	224	110010011000111
40	01111	104	1011	168	11001000	232	11001001100000
48	0101	112	1010	176	1100100111	240	11001001100001
56	1001	120	1000	184	11001001101		

Table 8.B.3 – huff_sampbr

index	huff_sampbr	index	huff_sampbr	index	huff_sampbr
-240	111111110110001000010	-72	111101101	96	11111111011100011
-232	111111110110001000011	-64	11110111	104	1111111110110101
-224	111111110110001000100	-56	1111010	112	111111110110000
-216	111111110110001000101	-48	111100	120	11111111011000101000 1
-208	111111110110001000110	-40	111110	128	11111111011000101001 0
-200	111111110110001000111	-32	11101	136	11111111011000101001 1
-192	111111110110001001000	-24	0111	144	11111111011000101010 0
-184	111111110110001001001	-16	010	152	11111111011000101010 1
-176	111111110110001001010	-8	00	160	11111111011000101011 0
-168	111111110110001001011	0	10	168	11111111011000101011 1
-160	111111110110001001100	8	110	176	11111111011000101100 0
-152	111111110110001001101	16	0110	184	11111111011000101100 1
-144	111111110110001001110	24	11100	192	11111111011000101101 0
-136	111111110110001001111	32	1111110	200	11111111011000101101 1
-128	111111110110001010000	40	11111110	208	11111111011000101110 0
-120	111111110110100	48	111101100	216	11111111011000101110 1
-112	11111111011001	56	1111111110	224	11111111011000101111 0
-104	1111111111100	64	11111111010	232	11111111011000101111 1
-96	111111110111	72	111111111111	240	11111111011000100000
-88	11111111110	80	1111111111101		
-80	1111111100	88	11111111011011		

Table 8.B.4 – huff_sampca

index	huff_sampca	index	huff_sampca	index	huff_sampca
0	011101101011	88	000	176	0011001
8	01101100	96	1111	184	011011011
16	0011000	104	1110	192	0110110100
24	0110111	112	1100	200	011011010101
32	011010	120	1011	208	0110110101001
40	00111	128	1000	216	01101101010001
48	10011	136	0101	224	0110110101000010
56	0100	144	0010	232	0110110101000011
64	0111	152	10010	240	011011010100000
72	1010	160	01100		
80	1101	168	001101		

Table 8.B.5 – huff_sampcr[0]

index	huff_sampcr[0]	index	huff_sampcr[0]	index	huff_sampcr[0]
-26	111001010111000	-8	0110111	10	01101100
-25	01101101110	-7	001000	11	00011010
-24	1110010101111	-6	111000	12	011011010
-23	000111001011	-5	00101	13	000111010
-22	011010111110	-4	11101	14	1110010111
-21	111001010100	-3	0011	15	0110110110
-20	00011100100	-2	1111	16	0001110011
-19	01101011110	-1	110	17	11100101101
-18	11100101100	0	10	18	01101101111
-17	0001110110	1	010	19	00011101110
-16	0110101110	2	0111	20	111001010110
-15	1110010100	3	0000	21	111001010101
-14	000111000	4	01100	22	011010111111
-13	011010110	5	00010	23	000111001010
-12	00011011	6	001001	24	111001010111101
-11	01101010	7	1110011	25	00011101111
-10	11100100	8	0110100	26	111001010111001
-9	0001111	9	0001100		

Table 8.B.6 – huff_sampcr[1]

index	huff_sampcr[1]	index	huff_sampcr[1]	index	huff_sampcr[1]
-26	100111001011	-8	110110	10	11011111
-24	11011100010	-6	11010	12	10011101
-22	110111101110	-4	1100	14	110111001
-20	11011110110	-2	111	16	1101111010
-18	1101110000	0	0	18	1001110011
-16	100111000	2	101	20	11011100011
-14	110111100	4	1000	22	10011100100
-12	11011101	6	10010	24	110111101111
-10	1001111	8	100110	26	100111001010

Table 8.B.7 – huff_sampcr[2]

index	huff_sampcr[2]	index	huff_sampcr[2]	index	huff_sampcr[2]
-28	01011000101	-8	0100	12	0101110
-24	010110000	-4	00	16	01011010
-20	010110011	0	1	20	010110010
-16	01011011	4	011	24	0101100011
-12	0101111	8	01010	28	01011000100

Table 8.B.8 – huff_sampcr[3]

index	huff_sampcr[3]	index	huff_sampcr[3]	index	huff_sampcr[3]
-32	00010101	-8	01	16	00011
-24	000100	0	1	24	0001011
-16	0000	8	001	32	00010100

Table 8.B.9 – huff_sfreqba

index	huff_sfreqba	index	huff_sfreqba
0	101111110100101100100	1944	110001110
8	101111110100101100101	1952	111111110
16	101111110100101100110	1960	10000101
24	101111110100101100111	1968	00101110
32	101111110100101101000	1976	10000110
40	101111110100101101001	1984	00100011
48	101111110100101101010	1992	11100100
56	101111110100101101011	2000	10011010
64	101111110100101101100	2008	00101111
72	101111110100101101101	2016	111101100
80	101111110100101101110	2024	01111101
88	101111110100101101111	2032	01110100
96	101111110100101110000	2040	111011010
104	101111110100101110001	2048	10111011
112	101111110100101110010	2056	10011011
120	101111110100101110011	2064	10010000
128	101111110100101110100	2072	00110000
136	10111111010011	2080	10111100
144	101111110100101110101	2088	01011011
152	101111110101	2096	111011011
160	10111111011	2104	01001011
168	100010000	2112	10111101
176	001010010	2120	10001100
184	01000010	2128	00110001
192	101111111	2136	110001111
200	111001101	2144	01111110
208	00010001	2152	10010001
216	111001110	2160	10110011
224	111100001	2168	111011100
232	101010101	2176	00001110
240	010000011	2184	10010010
248	1111101001	2192	10111110
256	111100010	2200	10100110
264	111001111	2208	00110010
272	100111011	2216	00001111
280	010111011	2224	00011000
288	1111101010	2232	110101111
296	100010001	2240	110110000
304	0111000011	2248	111111111
312	001010011	2256	11111001
320	1111101011	2264	00111100
328	000100100	2272	00010000
336	000100101	2280	10100111
344	1111101100	2288	01101101
352	000100110	2296	01011100
360	110000000	2304	00000000
368	00110110	2312	00111101
376	010000110	2320	01001100
384	011101010	2328	00011001
392	011101011	2336	00100100
400	00010100	2344	110110001
408	010111100	2352	00111110
416	111100011	2360	00100101
424	110000001	2368	11001000
432	010111101	2376	00100110

440	011101100	2384	00000001
448	011101101	2392	100011101
456	1101001100	2400	110010010
464	1010101001	2408	00110011
472	100111100	2416	10000111
480	00011111	2424	011001011
488	010111110	2432	01111111
496	1111101101	2440	00000010
504	111010000	2448	101101000
512	110000010	2456	01001101
520	00000100	2464	111101101
528	00100000	2472	00100111
536	111010001	2480	101101001
544	111100100	2488	110010011
552	111010010	2496	111101110
560	110000011	2504	111000111
568	01000100	2512	101101010
576	10010101	2520	00111111
584	01010001	2528	100100110
592	10101001	2536	111011101
600	101010110	2544	010011100
608	111100101	2552	110110010
616	011101110	2560	101101011
624	00000101	2568	101001001
632	10001001	2576	01000000
640	01100000	2584	101101100
648	01100001	2592	011110011
656	101010111	2600	100000000
664	100111101	2608	110010100
672	111010011	2616	00011010
680	101011000	2624	01001111
688	10101101	2632	100100111
696	10000010	2640	111001010
704	110100111	2648	100000001
712	00000110	2656	100000010
720	100010100	2664	101101101
728	001010100	2672	110010101
736	110101000	2680	000101111
744	010111111	2688	001101000
752	01010010	2696	111001011
760	011000100	2704	1110011001
768	011101111	2712	1101100110
776	111100110	2720	011011100
784	110101001	2728	000110110
792	111110111	2736	111011110
800	110000100	2744	110010110
808	111010100	2752	111101111
816	011000101	2760	010011101
824	1010110010	2768	0001111011
832	111100111	2776	110010111
840	01110001	2784	010100000
848	11001110	2792	101101110
856	01110010	2800	101010000
864	011110000	2808	1110111110
872	110111001	2816	101010001
880	110101010	2824	100000011
888	101011100	2832	1110111111

896	00101011	2840	010100001
904	01000101	2848	100101000
912	11001111	2856	1111000000
920	01100011	2864	110011000
928	110111100	2872	110110100
936	011001000	2880	1010110011
944	011001001	2888	1101100111
952	011001010	2896	1111110001
960	110111101	2904	001101001
968	101011101	2912	1101101010
976	000100111	2920	11011010110
984	00110111	2928	110011001
992	01100110	2936	00101000
1000	10000011	2944	11011010111
1008	00111000	2952	011011101
1016	011110001	2960	001101010
1024	110000101	2968	1001010010
1032	101011110	2976	1101101100
1040	1111110000	2984	011011110
1048	100111110	2992	00110101100
1056	10111000	3000	000110111
1064	10111001	3008	101101111
1072	11010110	3016	1001010011
1080	11011101	3024	0101110101
1088	111010101	3032	110011010
1096	100111111	3040	1111000001
1104	101000000	3048	0000001100
1112	101011111	3056	1111101000
1120	110000110	3064	1001110100
1128	110111110	3072	01110000101
1136	00000111	3080	0110111110
1144	101000001	3088	110011011
1152	010000111	3096	00110101101
1160	101100000	3104	11011011010
1168	001010101	3112	1101101110
1176	110111111	3120	1011111100
1184	001110011	3128	0000001101
1192	111111001	3136	0110111111
1200	110000111	3144	000111100
1208	101000010	3152	10011101010
1216	111000000	3160	11011011011
1224	101100001	3168	11011011110
1232	101100010	3176	0011010111
1240	111000001	3184	0000001110
1248	111000010	3192	0111000000
1256	01010011	3200	11011011111
1264	00111010	3208	01000001000
1272	111000011	3216	1011111101001011110110
1280	1101001101	3224	110111000000
1288	111101000	3232	1101110001
1296	110101011	3240	10011101011
1304	111000100	3248	110111000001
1312	01010100	3256	1110011000
1320	111111010	3264	0000001111
1328	11010000	3272	0111000001
1336	01100111	3280	1011111101001011110111
1344	01101000	3288	110111000010

1352	101000011	3296	010000010010
1360	00001000	3304	01000001010
1368	11000100	3312	010000010011
1376	10001011	3320	10101010000
1384	011110010	3328	0100000101100
1392	111101001	3336	0001111010
1400	10001100	3344	010000010111
1408	111101010	3352	101111110100101111000
1416	01010101	3360	0100000101101
1424	111000101	3368	110111000011
1432	00001001	3376	10101010001
1440	111010110	3384	01011101000
1448	111111011	3392	010111010010
1456	10100010	3400	010111010011
1464	11111000	3408	1011111101000
1472	01101001	3416	101111110100101111001
1480	01010110	3424	011100001000
1488	01101010	3432	101111110100101111010
1496	10001101	3440	101111110100101111011
1504	01101011	3448	101111110100101111100
1512	110101110	3456	101111110100101111101
1520	10010110	3464	101111110100101111110
1528	10010111	3472	101111110100101111111
1536	01010111	3480	011100001001
1544	01000110	3488	101111110100100000000
1552	111000110	3496	101111110100100000001
1560	01101100	3504	101111110100100000010
1568	00001010	3512	101111110100100000011
1576	01111010	3520	101111110100100000100
1584	00101100	3528	101111110100100000101
1592	111101011	3536	101111110100100000110
1600	00111011	3544	101111110100100000111
1608	01110011	3552	101111110100100001000
1616	01111011	3560	101111110100100001001
1624	11010001	3568	101111110100100001010
1632	111111100	3576	101111110100100001011
1640	100010101	3584	101111110100100001100
1648	01011000	3592	101111110100100001101
1656	10011000	3600	101111110100100001110
1664	100011100	3608	101111110100100001111
1672	10100011	3616	101111110100100010000
1680	110001010	3624	101111110100100010001
1688	0001110	3632	101111110100100010010
1696	00010101	3640	101111110100100010011
1704	10110010	3648	101111110100100010100
1712	111010111	3656	101111110100100010101
1720	01011001	3664	101111110100100010110
1728	00100001	3672	101111110100100010111
1736	00001011	3680	101111110100100011000
1744	00100010	3688	101111110100100011001
1752	01000111	3696	101111110100100011010
1760	00001100	3704	101111110100100011011
1768	111011000	3712	101111110100100011100
1776	11000110	3720	101111110100100011101
1784	11010010	3728	101111110100100011110
1792	00010110	3736	101111110100100011111
1800	01001000	3744	101111110100100100000

1808	101001000	3752	10111111010010100001
1816	110001011	3760	10111111010010100010
1824	01001001	3768	10111111010010100011
1832	10111010	3776	10111111010010100100
1840	01111100	3784	10111111010010100101
1848	000101110	3792	10111111010010100110
1856	00001101	3800	10111111010010100111
1864	10000100	3808	10111111010010101000
1872	01011010	3816	10111111010010101001
1880	00101101	3824	10111111010010101010
1888	01001010	3832	10111111010010101011
1896	111011001	3840	10111111010010101100
1904	111111101	3848	10111111010010101101
1912	10001111	3856	10111111010010101110
1920	10011001	3864	10111111010010101111
1928	10100101	3872	10111111010010110000
1936	101100011	3880	10111111010010110001

Table 8.B.10 – huff_sfreqbr

index	huff_sfreqbr	index	huff_sfreqbr
0	0000001011	1944	11000101001010
8	101111	1952	000011011111
16	10101	1960	000011100000
24	11011	1968	010100101101
32	11001	1976	111100101011
40	10100	1984	110001010011
48	10001	1992	010100101110
56	01100	2000	000011100001
64	01101	2008	000011100010
72	01011	2016	010100101111
80	00111	2024	000011100011
88	111101	2032	1100010101000
96	111110	2040	0111000000000
104	111010	2048	0111000000001
112	111001	2056	000011100100
120	100111	2064	010010000100011
128	101100	2072	000011100101
136	011111	2080	011100000001
144	100100	2088	100001111010
152	010101	2096	0111000000100
160	010000	2104	1100010101001
168	001100	2112	000011100110
176	001001	2120	1100010101010
184	000010	2128	000011100111
192	000001	2136	0111000000101
200	000111	2144	000011101000
208	1111110	2152	110001010010110
216	1111000	2160	000011101001
224	1100011	2168	0100100001000101111100010
232	1110111	2176	110001010010111
240	1101000	2184	000011101010
248	1001100	2192	110001010010110
256	1011010	2200	110001010010111
264	1100001	2208	0111000000110
272	1011100	2216	0111000000111
280	1000000	2224	1100010101100
288	1000010	2232	0111001101000
296	1001101	2240	1100010101101
304	0111101	2248	110001010111100
312	1001010	2256	0111001101001
320	0100010	2264	0111001101010
328	0111010	2272	000011101011
336	0100011	2280	110001010111101
344	0001011	2288	110001010111110
352	0100101	2296	0100100001000101111100011
360	0100110	2304	110001010111111
368	0001100	2312	0111001101011
376	0010100	2320	110001011000000
384	0011010	2328	110001011000001
392	0010000	2336	1100010110001
400	0011011	2344	1100010110010
408	0010101	2352	110001011000010
416	11111110	2360	0100100001000101111100100
424	0001000	2368	110001011000011
432	11101101	2376	110001011001110

440	11100000	2384	110001011001110
448	11010101	2392	010010000100010111100101
456	11100010	2400	010010000100010111100110
464	11010110	2408	11000101101000
472	11000001	2416	1000011110110
480	11101100	2424	11000101101001
488	10110110	2432	110001011001111
496	10111011	2440	11000101101010
504	10110111	2448	110001011010110
512	01001001	2456	1100010110110
520	00010011	2464	110001011010111
528	11010010	2472	110001011011100
536	01010011	2480	110001011011101
544	10000110	2488	110001011011110
552	01110110	2496	11000101110000
560	00101110	2504	010010000100010111100111
568	01001110	2512	110001011011111
576	01010000	2520	010010000100010111101000
584	01001111	2528	11000101110001
592	00101101	2536	010010000100010111101001
600	00000011	2544	010010000100010111101010
608	00000001	2552	1100010111001000
616	111100111	2560	010010000100010111101011
624	111100110	2568	1100010111001101
632	110100110	2576	010010000100010111101100
640	111100100	2584	010010000100010111101101
648	00001111	2592	1100010111001110
656	00101111	2600	1100010111001111
664	111111110	2608	1100010111010
672	101110101	2616	010010000100010111101110
680	110101110	2624	110001011101100
688	100101100	2632	010010000100010111101111
696	00010100	2640	110001011101101
704	110101111	2648	010010000100010111110000
712	111000011	2656	010010000100010111110001
720	011100011	2664	010010000100010111110010
728	110100111	2672	110001011101110
736	100000100	2680	110001011101111
744	100000101	2688	110001011110000
752	011101111	2696	110001011110001
760	011100001	2704	110001011110010
768	100001110	2712	11000101111010
776	101110100	2720	11000101111011
784	000110110	2728	110001011110011
792	011100100	2736	110001011111000
800	000100101	2744	010010000100010111110011
808	001011001	2752	010010000100010111110100
816	1111111110	2760	010010000100010111110101
824	1111001011	2768	11000101111101
832	100101101	2776	010010000100010111110110
840	010010001	2784	110001011111001
848	001000101	2792	010010000100010111110111
856	010100010	2800	010010000100010111111000
864	1100000001	2808	110001011111100
872	011110010	2816	010010000100010111111001
880	1001011101	2824	110001011111101
888	000110111	2832	010010000100010111111010

896	000101010	2840	11000101111111
904	1110001100	2848	0100100001000101111111011
912	000000100	2856	0100100001000101111111100
920	000011000	2864	111000111000000
928	1101010011	2872	0100100001000101111111101
936	1110000100	2880	111000111000001
944	000100100	2888	111000111000010
952	011100010	2896	0100100001000101111111110
960	1110001101	2904	0100100001000101111111111
968	0111100000	2912	0100100001000100000000
976	000101011	2920	11100011100010
984	0111100001	2928	0100100001000100000001
992	1101010000	2936	111000111000011
1000	0111000001	2944	111000111000110
1008	1101010001	2952	0100100001000100000010
1016	0111001010	2960	111000111000111
1024	0111001011	2968	00000010100000
1032	1000011111	2976	0100100001000100000011
1040	1100000010	2984	0100100001000100000100
1048	0010001001	2992	0100100001000100000101
1056	1101010010	3000	0100100001000100000110
1064	1100000000	3008	0100100001000100000111
1072	0111001110	3016	0100100001000100001000
1080	1110001111	3024	00000010100001
1088	11111111110	3032	0100100001000100001001
1096	0000111011	3040	0100100001000100001010
1104	0111100010	3048	0100100001000100001011
1112	0111100011	3056	0100100001000100001100
1120	0001101000	3064	0100100001000100001101
1128	0111001100	3072	00000010100010
1136	0111100111	3080	0100100001000100001110
1144	1001011110	3088	0100100001000100001111
1152	0010001000	3096	0100100001000100010000
1160	1100000001100	3104	00000010100011
1168	0111001111	3112	0100100001000100010001
1176	11111111111	3120	01001000010000
1184	0111011100	3128	0100100001000100010010
1192	0010001101	3136	0100100001000100010011
1200	1000001100	3144	0100100001000100010100
1208	0010001110	3152	0100100001000100010101
1216	0010110001	3160	0100100001000100010110
1224	0111011101	3168	0100100001000100010111
1232	10010111110	3176	0100100001000100011000
1240	0100100000	3184	0100100001000100011001
1248	01111001101	3192	0100100001000100011010
1256	0000000000	3200	0100100001000100011011
1264	1001011100	3208	0100100001000100011100
1272	0001101001	3216	0100100001000100011101
1280	0000000001	3224	0100100001000100011110
1288	0000000010	3232	0100100001000100011111
1296	01001000011	3240	0100100001000100100000
1304	11100001010	3248	0100100001000100100001
1312	110000001101	3256	0100100001000100100010
1320	0001101010	3264	0100100001000100100011
1328	0000000011	3272	0100100001000100100100
1336	0001101011	3280	0100100001000100100101
1344	01110011011	3288	0100100001000100100110

1352	10010111111	3296	0100100001000100100111
1360	01010001100	3304	0100100001000100101000
1368	11100011101	3312	0100100001000100101001
1376	01010001101	3320	0100100001000100101010
1384	110000001110	3328	0100100001000100101011
1392	1000011110111	3336	0100100001000100101100
1400	11100001011	3344	0100100001000100101101
1408	11000100000	3352	0100100001000100101110
1416	01111001100	3360	0100100001000100101111
1424	00100011110	3368	0100100001000100110000
1432	0010001000	3376	0100100001000100110001
1440	11000100001	3384	0100100001000100110010
1448	11000100010	3392	0100100001000100110011
1456	01010001110	3400	0100100001000100110100
1464	01010001111	3408	0100100001000100110101
1472	0000110010	3416	0100100001000100110110
1480	111000111001	3424	0100100001000100110111
1488	00100011111	3432	0100100001000100111000
1496	10000011010	3440	0100100001000100111001
1504	11000100011	3448	0100100001000100111010
1512	00000010101	3456	0100100001000100111011
1520	10000011011	3464	0100100001000100111100
1528	111100101000	3472	0100100001000100111101
1536	10000011100	3480	0100100001000100111110
1544	01010010000	3488	0100100001000100111111
1552	010010000101	3496	0100100001000101000000
1560	01010010001	3504	0100100001000101000001
1568	110000001111	3512	0100100001000101000010
1576	00001100110	3520	0100100001000101000011
1584	0100100001001	3528	0100100001000101000100
1592	0101001001000	3536	0100100001000101000101
1600	110001001000	3544	0100100001000101000110
1608	00001100111	3552	0100100001000101000111
1616	100000111010	3560	0100100001000101001000
1624	1100010010010	3568	0100100001000101001001
1632	110001001010	3576	0100100001000101001010
1640	00101100000	3584	0100100001000101001011
1648	010100100101	3592	0100100001000101001100
1656	00001101000	3600	0100100001000101001101
1664	100000111011	3608	0100100001000101001110
1672	01010010011	3616	0100100001000101001111
1680	10000011110	3624	0100100001000101010000
1688	00101100001	3632	0100100001000101010001
1696	00001101001	3640	0100100001000101010010
1704	110001001011	3648	0100100001000101010011
1712	11000100110	3656	0100100001000101010100
1720	110001001110	3664	0100100001000101010101
1728	110001001111	3672	0100100001000101010110
1736	111100101001	3680	0100100001000101010111
1744	100000111110	3688	0100100001000101011000
1752	00001101010	3696	0100100001000101011001
1760	1100010010011	3704	0100100001000101011010
1768	00001101011	3712	0100100001000101011011
1776	00001101100	3720	0100100001000101011100
1784	00001101101	3728	0100100001000101011101
1792	110001010000	3736	0100100001000101011110
1800	100000111111	3744	0100100001000101011111

1808	000000101001	3752	0100100001000101100000
1816	1100010100010	3760	0100100001000101100001
1824	100001111000	3768	0100100001000101100010
1832	0101001001001	3776	0100100001000101100011
1840	010100101000	3784	0100100001000101100100
1848	111100101010	3792	0100100001000101100101
1856	000011011100	3800	0100100001000101100110
1864	010100101001	3808	0100100001000101100111
1872	0101001010100	3816	0100100001000101101000
1880	100001111001	3824	0100100001000101101001
1888	1100010100011	3832	0100100001000101101010
1896	010100101011	3840	0100100001000101101011
1904	000011011101	3848	0100100001000101101100
1912	1100010100100	3856	0100100001000101101101
1920	010100101100	3864	0100100001000101101110
1928	0101001010101	3872	0100100001000101101111
1936	000011011110	3880	0100100001000101110000

Table 8.B.11 – huff_sfreqc

index	huff_sfreqc	index	huff_sfreqc
0	00100011011100011010010000	1944	011101111
8	00100011011100011010010001	1952	11101110
16	00100011011100011010010010	1960	10011100
24	00100011011100011010010011	1968	11010011
32	00100011011100011010010100	1976	00011011
40	00100011011100011010010101	1984	110010111
48	00100011011100011010010110	1992	111110001
56	00100011011100011010010111	2000	11001110
64	00100011011100011010011000	2008	10011001
72	00100011011100011010011001	2016	111111100
80	0010001101110011011	2024	00001100
88	00100011011100011010011010	2032	111010100
96	001000110111001100	2040	01000101
104	0010001101110010	2048	11110000
112	00100011011100111	2056	0100111
120	100110101011010	2064	00001110
128	100110101011000	2072	01000010
136	100110101011001	2080	111111101
144	0010001101100	2088	111101100
152	0010001101111	2096	00011100
160	011001001001	2104	01110101
168	111101000111	2112	01100000
176	11000000000	2120	10001111
184	0111000010	2128	10111110
192	0010011010	2136	10010101
200	11100010010	2144	1010010
208	0000101111	2152	11100100
216	0000001100	2160	01010111
224	10110110111	2168	00101010
232	11101011000	2176	01110011
240	11011100010	2184	10101011
248	10111101000	2192	11011000
256	10101001000	2200	01111101
264	01101011101	2208	00101011
272	10001011101	2216	00110101
280	11001101101	2224	11000011
288	0101100011	2232	10001001
296	0010111100	2240	111100110
304	0000001000	2248	111100010
312	11000000001	2256	10010100
320	01100100101	2264	11010111
328	10011010001	2272	11011101
336	0001100010	2280	11111111
344	001010000	2288	10111000
352	110001001	2296	10110111
360	01101011100	2304	11000111
368	10101110100	2312	01100110
376	0000000001	2320	11010101
384	1000001101	2328	10110011
392	0111000011	2336	111101010
400	0011101010	2344	110111100
408	000110010	2352	01001010
416	1111011110	2360	10000000
424	11101101001	2368	1010110
432	01101000000	2376	0001001

440	10001010001	2384	01010101
448	111101000110	2392	01010011
456	00100011010	2400	01111001
464	11011100000	2408	10010110
472	0000001101	2416	10101000
480	1001001011	2424	01011101
488	11100111000	2432	11010001
496	01101000001	2440	11001100
504	11101011001	2448	00100000
512	1011001011	2456	10111010
520	1101010011	2464	0001111
528	1100010000	2472	11101000
536	0000100010	2480	00101111
544	0000101110	2488	01110110
552	0110101101	2496	11100101
560	1001110111	2504	10101010
568	001110100	2512	10011111
576	011100011	2520	11111011
584	001000111	2528	10001101
592	110000011	2536	10000111
600	111111001	2544	11101100
608	10110110110	2552	0100000
616	11110100010	2560	01000011
624	1000001100	2568	111101011
632	011010010	2576	111101001
640	100001010	2584	00111101
648	100100110	2592	10101111
656	1010001011	2600	0010110
664	0000100011	2608	11101001
672	110011111111	2616	10011000
680	1100000001	2624	11001001
688	01011100	2632	10110100
696	00000111	2640	01010001
704	110001101	2648	10100110
712	01001001111	2656	10100011
720	1010100101	2664	00110110
728	001100011	2672	10001100
736	0100100110	2680	01111111
744	0101000010	2688	01001011
752	0010100000	2696	0111101
760	000110000	2704	11010000
768	011111100	2712	101111011
776	110000001	2720	110110101
784	00110111	2728	01011010
792	00000100	2736	10001000
800	11001111110	2744	11000010
808	10001010000	2752	11100001
816	11011100001	2760	10100000
824	11100010011	2768	01011001
832	111100011	2776	111110011
840	100001011	2784	10010111
848	111000110	2792	00111000
856	1011101111	2800	01011110
864	11010100100	2808	000010110
872	0110101100	2816	110010100
880	0110010011	2824	11100110
888	1001001010	2832	01100111

896	1111011111	2840	111010111
904	100100000	2848	00001101
912	00111100	2856	111110100
920	1100111110	2864	110101000
928	1000101111	2872	110101101
936	1011001010	2880	11100000
944	00010100	2888	111101110
952	1111011010	2896	00010101
960	10100010100	2904	00101001
968	10101110101	2912	011011111
976	11001101000	2920	011100000
984	0001100011	2928	10000001
992	000110011	2936	10111111
1000	010110000	2944	000000101
1008	100000111	2952	101111000
1016	010101000	2960	01100011
1024	0110101111	2968	111110010
1032	10111101001	2976	111010101
1040	11101101000	2984	10100001
1048	11001101100	2992	101101100
1056	0111000100	3000	01000100
1064	011101110	3008	110010000
1072	00110011	3016	100010110
1080	01111000	3024	00100001
1088	1001110110	3032	0011111
1096	11011100011	3040	01011011
1104	0101000011	3048	011111101
1112	1011110101	3056	01101110
1120	1111110000	3064	01010110
1128	1101100111	3072	10111001
1136	1110001000	3080	111000111
1144	011100100	3088	10110000
1152	01001101	3096	110011110
1160	011001000	3104	111110101
1168	0111001010	3112	111110000
1176	1000101001	3120	00100111
1184	110110010	3128	01001000
1192	10110001	3136	011000100
1200	0111001011	3144	101111001
1208	10011010000	3152	00100010
1216	00111010111	3160	01110100
1224	10100010101	3168	110010101
1232	11100111001	3176	101001111
1240	1100100011	3184	00100100
1248	101001110	3192	01100101
1256	100111010	3200	101101011
1264	111100111	3208	101010011
1272	001011101	3216	110101100
1280	1100010001	3224	001100010
1288	1111011011	3232	000010100
1296	1010111001	3240	1110101101
1304	0011001010	3248	1001101001
1312	0111000101	3256	1101100110
1320	001001100	3264	0010001100
1328	011011110	3272	1100100010
1336	11010010	3280	0000001001
1344	00001111	3288	000000111

1352	1010111000	3296	10101001001
1360	000110100	3304	10011010100
1368	010101001	3312	0110100001
1376	110111111	3320	00000110
1384	01000110	3328	0000101011
1392	1001101011	3336	1011101110
1400	1110011101	3344	0000000000
1408	010010010	3352	0011001011
1416	011010001	3360	11010100101
1424	101000100	3368	10001011100
1432	001010001	3376	1100110111
1440	011000101	3384	1011011010
1448	101110110	3392	11001101001
1456	10011110	3400	00001010101
1464	10010001	3408	00001010100
1472	000010000	3416	0010001101101
1480	1010111011	3424	0101100010
1488	1100110101	3432	0010011011
1496	1111110001	3440	0110010010000
1504	010111111	3448	0010001110111000
1512	1111010000	3456	100110101101110
1520	111100101	3464	100110101101111
1528	111100100	3472	1001101010101
1536	0001011	3480	00111010110
1544	01001100	3488	001000110111010
1552	000110101	3496	1001101010100
1560	1110110101	3504	0010001101110011010011011
1568	010111110	3512	0010001101110011010011100
1576	101100100	3520	1001101010110111
1584	110010110	3528	001000110111011
1592	100100111	3536	0010001101110011010011101
1600	111011011	3544	1001101010110110
1608	00001001	3552	0010001101110011010011110
1616	01101010	3560	0010001101110011010011111
1624	11111101	3568	0010001101110011010100000
1632	111001111	3576	0010001101110011010100001
1640	000000001	3584	0010001101110011010100010
1648	00111001	3592	0010001101110011010100011
1656	00100101	3600	0010001101110011010100100
1664	11000101	3608	0010001101110011010100101
1672	10001110	3616	0010001101110011010100110
1680	001100100	3624	0010001101110011010100111
1688	110110100	3632	0010001101110011010101000
1696	01100001	3640	0010001101110011010101001
1704	111011110	3648	0010001101110011010101010
1712	100100100	3656	0010001101110011010101011
1720	101101010	3664	0010001101110011010101100
1728	111000101	3672	0010001101110011010101101
1736	00011101	3680	0010001101110011010101110
1744	01111100	3688	0010001101110011010101111
1752	10000100	3696	0010001101110011010110000
1760	00000001	3704	0010001101110011010110001
1768	00000101	3712	0010001101110011010110010
1776	111011111	3720	0010001101110011010110011
1784	00111011	3728	0010001101110011010110100
1792	110000010	3736	0010001101110011010110101
1800	110001100	3744	0010001101110011010110110

1808	100010101	3752	00100011011100110101101111
1816	00010000	3760	00100011011100110101110000
1824	00010001	3768	00100011011100110101110001
1832	0110110	3776	00100011011100110101110010
1840	10000110	3784	00100011011100110101110011
1848	100100001	3792	00100011011100110101110100
1856	011010011	3800	00100011011100110101110101
1864	01000111	3808	00100011011100110101110110
1872	00110000	3816	00100011011100110101111111
1880	10000010	3824	00100011011100110100000000
1888	00110100	3832	00100011011100110100000001
1896	110111101	3840	00100011011100110100000010
1904	110111001	3848	00100011011100110100000011
1912	01010010	3856	00100011011100110100000100
1920	10011011	3864	00100011011100110100000101
1928	11011011	3872	00100011011100110100000110
1936	110111110	3880	00100011011100110100000111

Table 8.B.12 – huff_nlag

index	huff_nlag	index	huff_nlag
-512	100010101101101100	4	110
-508	100010101101101101	8	1111
-504	100010101101101110	12	10010
-500	100010101101101111	16	100001
-496	100010111010000000	20	1000100
-492	100010111010000001	24	10000000
-488	100010111010000010	28	100000010
-484	100010111010000011	32	1000101000
-480	100010111010000100	36	10001010011
-476	100010111010000101	40	100010111001
-472	100010111010000110	44	100010101110
-468	100010111010000111	48	1000101011010
-464	100010111010001000	52	1000101011110
-460	100010111010001001	56	100010101101111
-456	100010111010001010	60	1000101001000
-452	100010111010001011	64	100010101101001
-448	100010111010001100	68	100010101111100
-444	100010111010001101	72	10001010010010
-440	100010111010001110	76	100010111011110
-436	100010111010001111	80	100010100101110
-432	100010111010010000	84	100010111101110110
-428	100010111010010001	88	100010101111101
-424	100010111010010010	92	100010100101111
-420	100010111010010011	96	1000101110111110
-416	100010111010010100	100	100010111011101111
-412	100010111010010101	104	1000101110111111
-408	100010111010010110	108	1000101011011000
-404	100010111010010111	112	1000101110111001111
-400	100010111010011000	116	1000101011011001
-396	100010111010011001	120	10000001100000000
-392	100010111010011010	124	1000101011011010
-388	100010111010011011	128	10000001100000001
-384	100010111010011100	132	10000001100000010
-380	100010111010011101	136	10000001100000011
-376	100010111010011110	140	10000001100000100
-372	100010111010011111	144	10000001100000101
-368	100010111010100000	148	10000001100000110
-364	100010111010100001	152	10000001100000111
-360	100010111010100010	156	10000001100001000
-356	100010111010100011	160	10000001100001001
-352	100010111010100100	164	1000000110000101
-348	100010111010100101	168	10000001100001100
-344	100010111010100110	172	10000001100001101
-340	100010111010100111	176	1000000110000111
-336	100010111010101000	180	10000001100010000
-332	100010111010101001	184	10000001100010001
-328	100010111010101010	188	10000001100010010
-324	100010111010101011	192	1000000110001010
-320	100010111010101100	196	10000001100010011
-316	100010111010101101	200	10000001100010110
-312	100010111010101110	204	10000001100010111
-308	100010111010101111	208	10000001100011000
-304	100010111010110000	212	10000001100011001
-300	100010111010110001	216	10000001100011010
-296	100010111010110010	220	1000000110001110

-292	100010111010110011	224	10000001100011011
-288	100010111010110100	228	10000001100011110
-284	100010111010110101	232	10000001100011111
-280	100010111010110110	236	10000001100100000
-276	100010111010110111	240	10000001100100001
-272	100010111010111000	244	10000001100100010
-268	100010111010111001	248	10000001100100011
-264	100010111010111010	252	10000001100100100
-260	100010111010111011	256	10000001100100101
-256	100010111010111100	260	10000001100100110
-252	100010111010111101	264	10000001100100111
-248	100010111010111110	268	10000001100101000
-244	100010111010111111	272	10000001100101001
-240	100010111011000000	276	10000001100101010
-236	100010111011000001	280	10000001100101011
-232	100010111011000010	284	10000001100101100
-228	100010111011000011	288	10000001100101101
-224	100010111011000100	292	10000001100101110
-220	100010111011000101	296	10000001100101111
-216	100010111011000110	300	10000001100110000
-212	100010111011000111	304	10000001100110001
-208	100010111011001000	308	10000001100110010
-204	100010111011001001	312	10000001100110011
-200	10001011101100101	316	10000001100110100
-196	100010111011001100	320	10000001100110101
-192	100010111011001101	324	10000001100110110
-188	100010111011001110	328	10000001100110111
-184	100010111011001111	332	10000001100111000
-180	100010111011010000	336	10000001100111001
-176	100010111011010001	340	10000001100111010
-172	100010111011010010	344	10000001100111011
-168	100010111011010011	348	10000001100111100
-164	100010111011010100	352	10000001100111101
-160	10001011101101011	356	10000001100111110
-156	10001010011011001	360	10000001100111111
-152	100010111011010101	364	10000001101000000
-148	100010111011011000	368	10000001101000001
-144	100010111011011001	372	10000001101000010
-140	100010111011011010	376	10000001101000011
-136	100010111011011011	380	10000001101000100
-132	10001011101101110	384	10000001101000101
-128	10001011101101111	388	10000001101000110
-124	100010111011100000	392	10000001101000111
-120	100010111011100001	396	10000001101001000
-116	100010111011100010	400	10000001101001001
-112	100010111011100011	404	10000001101001010
-108	100010111011100100	408	10000001101001011
-104	100010111011100101	412	10000001101001100
-100	100010111011100110	416	10000001101001101
-96	1000101001011010	420	10000001101001110
-92	100010100100111	424	10000001101001111
-88	1000101001011011	428	10000001101010000
-84	1000101110111010	432	10000001101010001
-80	10001010010100	436	10000001101010010
-76	10001010010101	440	10000001101010011
-72	10001010110010	444	10000001101010100
-68	10000001101100	448	10000001101010101

-64	10000001101101	452	10000001101010110
-60	10001010111111	456	10000001101010111
-56	1000000110111	460	10000001101011000
-52	1000101011000	464	10000001101011001
-48	100010101010	468	10000001101011010
-44	100010101011	472	10000001101011011
-40	100010111000	476	10000001101011100
-36	10001010100	480	10000001101011101
-32	1000000111	484	10000001101011110
-28	1000101111	488	10000001101011111
-24	100010110	492	10001010010011000
-20	1000001	496	10001010010011001
-16	100011	500	10001010010011010
-12	10011	504	10001010010011011
-8	1110	508	10001010010110000
-4	101	512	10001010010110001
0	0		

Table 8.B.13 – huff_nlsf

index	huff_nlsf	index	huff_nlsf	index	huff_nlsf
7	10110011	14	1000	21	1011010
8	101101110	15	00	22	10010011
9	101101111	16	11	23	100100101
10	10110010	17	01	24	100100100
11	1001000	18	1010	25	10110110
12	1011000	19	10111		
13	100101	20	10011		

Note: Values smaller than value 8 are represented by 'escape' value 7 followed by a 3 bits unsigned integer containing the real value. Values greater than value 24 are represented by 'escape' value 25 followed by an 8 bits unsigned integer containing the real value.

Table 8.B.14 – huff_ngain

index	huff_ngain	index	huff_ngain	index	huff_ngain
-13	010011100	-4	0100010	5	01000111
-12	010011011001	-3	010010	6	010011001
-11	010011101110	-2	01011	7	0100110111
-10	010011101011	-1	00	8	0100110100
-9	01001110100	0	1	9	01001101101
-8	01001110110	1	011	10	010011101111
-7	0100110101	2	01010	11	010011101010
-6	010011000	3	010000	12	010011011000
-5	01000110	4	01001111		

Note: Values outside the range [-12 ..12] are represented by 'escape' value –13 followed by an 8 bits signed integer containing the real value.

Table 8.B.15 – huff_scont

index	huff_scont	index	huff_scont	index	huff_scont
0	1000	4	001	8	0000
1	0100	5	1001	9	11
2	101	6	0101		
3	011	7	0001		

Table 8.B.16 – huff_nrofbirths

Index	huff_nrofbirths	Index	huff_nrofbirths	Index	huff_nrofbirths
0	010	21	000001001	42	000000100001110
1	1010	22	00000101	43	000000100001111
2	1110	23	000001100	44	000000100010000
3	011	24	00000001	45	000000100010001
4	110	25	0000010001	46	000000100010010
5	100	26	0000011010	47	000000100010011
6	001	27	0000011011	48	000000100010100
7	1111	28	000000100000100	49	000000100010101
8	0001	29	0000011100	50	000000100010110
9	10111	30	000000100000101	51	000000100010111
10	10110	31	000000100000110	52	000000100011000
11	0000111	32	000000100000111	53	000000100011001
12	000010	33	0000011101	54	000000100011010
13	0000110	34	000000100001000	55	000000100011011
14	00000011	35	0000011110	56	000000100011100
15	00000000	36	000000100001001	57	000000100011101
16	000000101	37	000000100001010	58	000000100011110
17	0000001001	38	0000011111	59	000000100011111
18	0000010000	39	000000100001011	60	000000010000000
19	000000100000010	40	000000100001100		
20	000000100000011	41	000000100001101		

Table 8.B.17 – huff_iid_df[0] and huff_iid_dt[0]

Index	huff_iid_df[0]	huff_iid_dt[0]	Index	huff_iid_df[0]	huff_iid_dt[0]
-30	011111111010110100	0100111011010100	1	010	00
-29	011111111010110101	0100111011010101	2	0001	01011
-28	011111110101110110	0100111011001110	3	01101	010010
-27	011111110101110111	0100111011001111	4	011101	0100001
-26	011111110101110100	0100111011001100	5	0111101	01001100
-25	011111110101110101	0100111011010110	6	01111101	010011011
-24	011111111010001010	0100111011011000	7	011111100	01001110010
-23	011111111010001011	0100111101000110	8	0111111100	01001111001
-22	011111111010001000	0100111101100000	9	01111111100	01001110000
-21	01111111010000000	010011100011000	10	01111110100	010011101111
-20	011111111010110110	010011100011001	11	011111101011	010011100010
-19	01111111010000010	010011101100100	12	0111111101010	0100111101010
-18	01111111010111000	010011101100101	13	01111111101010	0100111011000
-17	0111111101000010	010011101101101	14	011111111010110	01001111010111
-16	0111111110101110	010011110110001	15	011111111010000	01001111010000
-15	011111110101111	01001110110111	16	0111111110101111	010011110110010
-14	01111111010001	01001111010110	17	0111111101000011	010011110100010
-13	01111111101001	0100111000111	18	011111110101111001	010011100011010
-12	0111111101001	0100111101001	19	011111110100000011	010011100011011
-11	011111101010	0100111101101	20	011111111010110111	0100111101100110
-10	011111111011	010011101110	21	011111110100000001	0100111101100111
-9	01111111011	010011110111	22	011111111010001001	0100111101100001
-8	0111111011	01001111000	23	011111111010001110	0100111101000111
-7	0111111111	0100111001	24	011111111010001111	0100111011011001
-6	01111100	010011010	25	011111111010001100	0100111011010111
-5	0111100	010011111	26	011111111010001101	0100111011001101
-4	011100	0100000	27	011111111010110010	0100111011010010
-3	01100	010001	28	011111111010110011	0100111011010011
-2	0000	01010	29	011111111010110000	0100111011010000
-1	001	011	30	011111111010110001	0100111011010001
0	1	1			